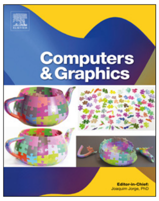
Computers & Graphics 115 (2023) 122–136

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag



Special Section on 3DOR 2023

MARF: The Medial Atom Ray Field object representation

Peder Bergebakken Sundt [*], Theoharis Theoharis

Norwegian University of Science and Technology, Norway

ARTICLE INFO

Article history:
Received 31 March 2023
Received in revised form 3 June 2023
Accepted 25 June 2023
Available online 28 June 2023

Keywords:
Representation learning
Neural fields
Differential geometry
Medial Axis Transform

ABSTRACT

We propose Medial Atom Ray Fields (MARFs), a novel neural object representation that enables accurate differentiable surface rendering with a single network evaluation per camera ray. Existing neural ray fields struggle with multi-view consistency and representing surface discontinuities. MARFs address both using a medial shape representation, a dual representation of solid geometry that yields cheap geometrically grounded surface normals, in turn enabling computing analytical curvature despite the network having no second derivative. MARFs map a camera ray to multiple medial intersection candidates, subject to ray-sphere intersection testing. We illustrate how the learned medial shape quantities applies to sub-surface scattering, part segmentation, and aid representing a space of articulated shapes. Able to learn a space of shape priors, MARFs may prove useful for tasks like shape retrieval and shape completion, among others. Code and data can be found at https://github.com/pbsds/MARF.

1. Introduction

Learning efficient and accurate ways to represent 3D geometry is valuable to applications such as 3D shape analysis, computer graphics, computer vision, and robotics. The recent discovery of *neural fields*, also known as coordinate-based networks or implicit neural representations, has brought a renewed interest in visual computing problems. While simple in construction, neural fields exhibit an impressive ability to compactly represent, manipulate and generate continuous signals of arbitrary resolution and dimensionality across a plethora of modalities, in our case 3D geometry. They can also learn the underlying space of the training shapes, useful in applications such as generative shape modeling, shape infilling/completion, and shape retrieval.

While Cartesian neural fields represent 3D volumes admirably, rendering them requires ray-marching or sphere-tracing, where each sample along the ray in turn requires a full network evaluation which is expensive. In this paper we avoid sphere-tracing entirely, by parameterizing the field in terms of rays instead of points. Visualized in Fig. 1, we explore neural fields that map an oriented ray directly to its surface intersection point via an intermediate medial representation. This enables efficient real-time single-evaluation differentiable neural surface rendering and extraction.

We propose *Medial Atom Ray Fields* (MARFs), visualized in Fig. 2, which map oriented rays to a set of spherical intersection candidates called *medial atoms*, that are maximally inscribed in the represented shape pinned tangential to the ray-surface intersection point. From a MARF prediction, a simple line-sphere intersection test between the ray and the n predicted atoms is all one needs to jointly determine *where* and *whether* the ray hits. This medial representation also allows computing the surface normal without analytical network differentiation, which essentially means that we get it for free. This in turn enables computing the surface curvature, a second derivative quantity, despite the second derivative of our piecewise linear network being zero.

We identify two key challenges that hinder the usefulness of ray fields, which we address with the medial shape representation.

The first challenge is that ray fields are not by construction multi-view consistent like their 3D Cartesian counterparts. This is because the four Degrees of Freedom (DoF) of the input rays may cause a predicted 3D point to change appearance across views. Prior works focus on learning a latent manifold of sound ray fields. Our proposed MARF instead phrase the output domain in Cartesian space, which is stable w.r.t. change in incident viewing direction. We further enforce multi-view consistency during training with a novel multi-view loss.

The second challenge is for ray fields to represent discontinuities like sharp edges and overlapping geometry common to depth maps. Neural fields being Lipschitz continuous in their inputs [1,2] produce interpolation artifacts across such jumps. Prior works either sidestep the issue by relaxing the problem or use a filtering scheme to discard outliers. Our medial representation allows us to regularize multiple predictions to behave

[*] Corresponding author.
E-mail addresses: peder.b.sundt@ntnu.no (P.B. Sundt), theotheo@ntnu.no (T. Theoharis).

https://doi.org/10.1016/j.cag.2023.06.032

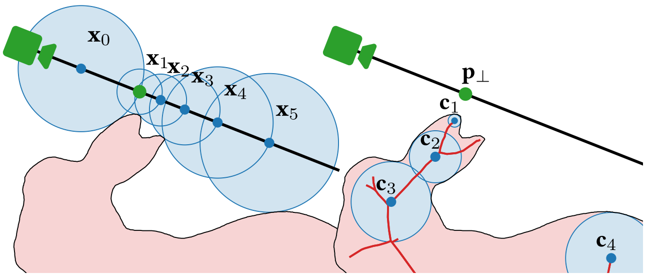


Fig. 1. A 2D slice of the Stanford bunny. On the left we sphere-trace it. On the right we show its medial axis, four medial atoms, and the projection $\mathbf{p}_\perp$ of nearest atom center $\mathbf{c}_1$ onto the line. Each tracing step requires an evaluation of the distance field which proves expensive when represented with a neural field. We explore MARFs which map the *line* to n medial intersection candidates in a single evaluation. The medial representation has many downstream use-cases.

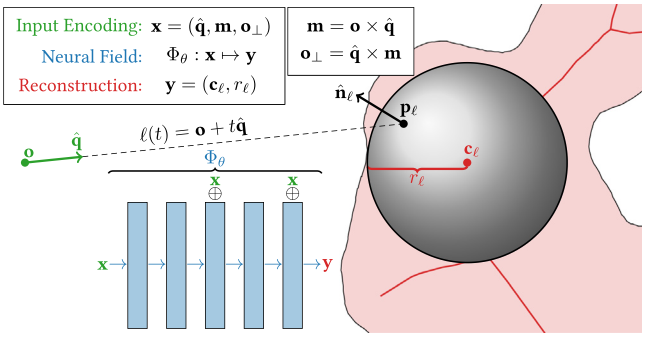

Input Encoding: $\mathbf{x} = (\hat{\mathbf{q}}, \mathbf{m}, \mathbf{o}_\perp)$
Neural Field: $\Phi_\theta : \mathbf{x} \mapsto \mathbf{y}$
Reconstruction: $\mathbf{y} = (\mathbf{c}_\ell, r_\ell)$
$\mathbf{m} = \mathbf{o} \times \hat{\mathbf{q}}$
$\mathbf{o}_\perp = \hat{\mathbf{q}} \times \mathbf{m}$
$\hat{\mathbf{n}}_\ell$
$\mathbf{p}_\ell$
$\ell(t) = \mathbf{o} + t\hat{\mathbf{q}}$
$\mathbf{o}$
$\hat{\mathbf{q}}$
Φ_θ
$\mathbf{x}$
$\oplus$
$\mathbf{x}$
$\oplus$
$\mathbf{c}_\ell$
r_ℓ
$\mathbf{x}$
$\mathbf{y}$

Fig. 2. A small MARF network Φ_θ illustrated. Given a ray ℓ with origin $\mathbf{o}$ and direction $\hat{\mathbf{q}}$ it predicts the maximally inscribed medial atom/sphere $(\mathbf{c}_\ell, r_\ell)$ whose intersection point $\mathbf{p}_\ell$ with the ray ℓ corresponds to the intersection between ℓ and the represented object (here shown with its medial axis in red). To uniquely encode rays we normalize $\hat{\mathbf{q}}$ and trade $\mathbf{o}$ for its moment $\mathbf{m}$ and perpendicular foot $\mathbf{o}_\perp$. The network is a simple MLP with skip connections, here illustrated with 4 hidden layers, where $\oplus$ denotes vector concatenation and → denotes a linear map.

well. We alleviate discontinuities by making each candidate specialize on different shape "limbs" while adhering to the medial constraints. This is achieved through a principled network initialization scheme and through regularization. Also, by labeling each ray hit by the candidate which produced it, a part segmentation emerges unsupervised.

The learned medial representation is of significant interest in 3D shape analysis, being applicable to classification, semantic manipulation, and segmentation. The represented medial axis, also known as the topological skeleton, produces smooth interpolations in the learned latent space of shapes. The medial radius, also known as the local feature size, is useful in shape analysis and visualization.

In summary, we make the following contributions:

- We propose learning MARFs, which map oriented rays to a set of medial atom intersection candidates that both classify rays as hit/miss and deliver the point of intersection as well as introduce a geometric 3D inductive bias, which oriented ray fields thus far have lacked.
- We demonstrate how MARFs allow computing the analytical surface curvature, despite the network being piecewise linear, using the medial atom normals.
- We introduce a multi-view consistency loss to constrain ray fields to generalize better from sparse set of training views.
- We show that MARFs can learn a space of shape priors.
- We show that MARFs discover a part segmentation unsupervised.

Scope. We target object-centric surface rendering. While one can compose multiple MARFs into a scene, we consider this outside the scope of this paper. We explore a global shape representation, where a single network represents the shapes without spatial partitioning.

In Section 2 we outline prior work and establish key preliminaries, in Section 3 we discuss our method, in Section 4 we evaluate our method, and in Section 5 we conclude our work and discuss future directions.

2. Background and related work

In this section we discuss related works while covering preliminaries about neural (ray) fields and the medial axis.

Representing 3D geometry and scenes with neural fields. Neural fields emerged in 2019 as a compact, continuous and flexible way to represent signals parametrized with spatial and temporal coordinates. The seminal papers [3–5] use them to represent a set of closed 3D shapes $\{\mathcal{O}_i \subset \mathbb{R}^3\}_{i=1}^n$ of arbitrary topology, either by learning their binary occupancy field or Signed Distance Field (SDF). The SDF $d^\pm_{\partial\mathcal{O}_i} : \mathbb{R}^3 \to \mathbb{R}$ in particular represents $\mathcal{O}_i$ by mapping 3D coordinates to the distance of the nearest surface boundary $\partial\mathcal{O}_i$, where interior distances are negative and exterior ones are positive:

$$d^\pm_{\partial\mathcal{O}_i}(\mathbf{x}) = \min_{\mathbf{x}' \in \partial\mathcal{O}_i} \left\|\mathbf{x} - \mathbf{x}'\right\| \cdot \begin{cases} 1 & \text{if } \mathbf{x} \notin \mathcal{O}_i \\ -1 & \text{if } \mathbf{x} \in \mathcal{O}_i \end{cases} \tag{1}$$

Neural fields may in fact learn any field $f : \mathcal{X} \to \mathcal{Y}$ that maps input coordinates $\mathbf{x} \in \mathcal{X}$ to signal values $\mathbf{y} \in \mathcal{Y}$, given enough $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ supervision examples. Examples include: Unsigned distance fields [6,7] which can represent non-watertight surfaces. Winding number fields [8] which can represent self-intersecting geometry. Closest surface point fields [7] which map to the nearest point on the surface ($\mathbb{R}^3 \to \partial\mathcal{O}$). Deep medial fields [9] which represent the local feature size of the nearest surface. Category-level shape descriptor fields, used in robotic manipulation [10].

Neural fields prove effective at learning complex mappings one would consider ill-posed to optimize [11,12]. Atzmon et al. [13] show this by learning the SDF without inside/outside supervision. This ability is thanks in part to over-parametrization, and in part to being analytically differentiable w.r.t. input coordinates [14]. Using double back-propagation one may fit neural fields to satisfy Partial Differential Equations (PDEs), or supervise the field gradient with sensor data, enabling data-driven discovery of PDEs [15–18]. For SDFs such a PDE is the Eikonal equation, which constrains the field gradient $\nabla_\mathbf{x} d^\pm_{\partial\mathcal{O}}(\mathbf{x})$ to be of unit length [16]. Its direction equals the normal vector near the surface boundary. Network differentiation is also useful during inference, as one may compute differential geometric quantities such as surface normals and curvature [19,20].

Neural fields excel in high-dimensional problems, since their size grows with target signal complexity instead of resolution. Mildenhall et al. [21] show this with NeRFs: a mapping with 5 Degrees of Freedom (DoF). NeRFs represent both the density and anisotropic (view-dependent) radiance field of 3D scenes. By ray marching these fields they achieve realistic novel-view synthesis from registered 2D image supervision. NeRF advancements and applications are plentiful [12,22,23], including adaptation to low-light High Dynamic Range scenarios [24,25], modeling complex materials [26] and registration [27]. Guo et al. [28] extend NeRF to consider the incident light direction, making for a 7 DoF mapping. Video NeRFs [29–32] go further by adding a temporal dimension, an axis along which both density and radiance may undergo extensive changes, showcasing impressive results.

For further details on neural fields, we encourage the reader to view the excellent overview of Xie et al. [12].

Speeding up rendering. A *forward map* relates (e.g. 3D volume) neural fields to domains where sensor data is available (e.g. 2D maps). In *volume rendering* the forward map may integrate the color contribution along rays cast through a 3D field [33], while in *surface rendering* it may locate the first ray-surface intersection or nearest distance [34,35]. One such forward map is ray-marching, which samples points equidistantly along the ray to numerically approximate the ray integral. This requires in the order of tens or hundreds of field evaluations, proving expensive with neural fields.

Several works address this problem which we divide into three categories:

The first category is making field evaluations cheaper. *Local* methods achieve this by subdividing the field into simple patches or chunks represented by smaller separate networks [36–42], while *hybrid* methods decode a grid of conditioning vectors with a simple decoder network [43–48]. Local and hybrid methods in effect bypass the difficulty of globally learning shapes with a single network. *Tabulation* methods forgo the neural network in favor of discrete data structures and interpolation [49–51]. Works in this category often achieve higher fidelity but are often unable to learn global shape priors. Some global methods *bake* their fields offline before rendering, where one essentially extracts a tabulation [52,53].

The second category seeks to algorithmically reduce the number of field evaluations, avoiding sampling empty or obscured regions. Surface rendering methods often opt to learn a distance field [5,9,54,55] such as the SDF which permits sphere-tracing [34] (visualized in Fig. 1). Volume rendering methods may perform a coarse pre-evaluation to produce an index [53,56,57], or construct a Monte-Carlo estimate of the ray integral [58].

The third category is our focus: directly predicting the ray integral, discussed in the next subsection.

Neural ray fields. To represent a ray field one must parametrize the domain of rays. Rays can be represented in a plethora of ways, the simplest being a tuple of two 3D points through which the ray passes. Front-facing neural light fields [59,60] prove with such a representation able to map rays to observed colors in highly realistic scenes. They consider rays cast between points on the near and far plane, which cannot represent 360° ray fields. The challenge is how to uniquely encode rays without symmetries.

A 3D line $\ell(t) = \mathbf{o} + t\hat{\mathbf{q}}$ parametrized by some origin $\mathbf{o}$ and direction $\hat{\mathbf{q}}$, has 4 DoF: compared to the 6 DoFs of 3D rigid bodies, lines lose two being invariant to translations along the line direction and rotations about the line axis. Rays technically gain a DoF over lines, featuring a starting point, but both we and the prior works discussed below discard this DoF and consider rays and lines equivalently. This effectively places the observer infinitely far away.

It is impossible to represent the space of rays/lines in a 4D vector space that is uniform and without singular directions, discontinuities or special cases [61–63]. Naively using the uniform 6D vector $(\mathbf{o}, \hat{\mathbf{q}})$ however produces a highly symmetric space that hinders learning.

Lindell et al. [18] learn segments of the volume rendering equation integral [33] by splitting the ray into n sections. They sample k points along each section and feed them along with ray direction $\hat{\mathbf{q}}$ into their network. While this ray parametrization enables the use of positional encoding [21,64], it is sensitive to the ray sampling positions and to the number of segments chosen. Mukund et al. [65] also sample k points along the ray, but associate each point with colors from multiple source views, then interpolate between them with a transformer model.

Neff et al. [66] accelerate rendering NeRFs by training an accompanying *oracle* network which predicts, given a ray, the salient segments along that ray to be further ray-marched. Yenamandra et al. [67] accelerate sphere-tracing neural SDFs by training an accompanying network to predict, given a ray, some initial starting point. They both represent ℓ as the 6D vector $(\mathbf{o}, \hat{\mathbf{q}})$, and remove two DoF by normalizing $\hat{\mathbf{q}}$ to be of unit size, and limit $\mathbf{o}$ to lie on the sphere (with fixed radius r) circumscribed around the reconstruction volume. This in effect restricts the 6D vector to a 4D subspace, or manifold, embedded in 6D: $(r^{-1}\mathbf{o}, \hat{\mathbf{q}}) \in S^2 \times S^2 \subset \mathbb{R}^6$ where S^2 is the unit 2-sphere. This representation has a finite reconstruction volume determined by r.

Sitzmann et al. [63] forgo the Cartesian radiance field and learn 360° neural light fields (LFN) directly. They represent rays using 6D Plücker coordinates [68]. Normalized Plücker coordinates encode the ray ℓ as $(\hat{\mathbf{q}}, \mathbf{m})$, where $\mathbf{m} = \mathbf{o} \times \hat{\mathbf{q}}$ is the *moment* vector of the ray origin $\mathbf{o}$ about the coordinate system origin. Plücker coordinates are thus restricted to $S^2 \times T^2_{\mathbf{o}} \subset \mathbb{R}^6$, where $T^n_{\mathbf{d}} = \{\mathbf{x} \in \mathbb{R}^{n+1} : \mathbf{x} \cdot \mathbf{d} = 0\}$ is the tangent space orthogonal to $\mathbf{d} \neq \mathbf{0}$, containing the coordinate system origin.

Feng et al. [69] target with PRIF surface rendering instead of light fields. They trade the moment $\mathbf{m}$ for the more geometrically grounded perpendicular foot $\mathbf{o}_\perp = \hat{\mathbf{q}} \times \mathbf{m}$, i.e. the orthogonal projection of the coordinate origin onto the ray ℓ. $(\hat{\mathbf{q}}, \mathbf{o}_\perp) \in S^2 \times T^2_{\mathbf{m}}$. We visualize both $\mathbf{m}$ and $\mathbf{o}_\perp$ in Fig. 3(a).

Both [67] and [69] compute the ray-surface intersection point by predicting the (signed) displacement along the ray from their normalized ray origin. This does not represent *whether* the ray intersects or not, which both works solve with a separate network output classifying hit/miss rays. While it does allow representing non-watertight geometry, it is an independent quantity that is not geometrically grounded, which generalizes poorly to unseen views.

Ray fields struggle to represent surface discontinuities, common near boundaries and overlapping geometry, due to neural networks being Lipschitz continuous on their inputs. The works of [66,67,69] all produce interpolation artifacts near surface discontinuities, and is addressed in two ways: [66,67] reduce the impact of discontinuities by relaxing the task to aid sampling a Cartesian network, and [69] opt to filter outlier predictions with high gradients. Neff et al. [66] note how multiple depth predictions do not improve their results.

An open problem for neural ray fields is *multi-view consistency*. Neural fields feature an inductive bias inherited from the structure of the input domain, and Cartesian fields find success thanks to this and to being multi-view consistent by construction. Ray fields do not share these qualities. Their extra DoF may cause a predicted point to change appearance across views. [63] address this with meta-learning [70–72], learning a latent space of light fields that are multi-view consistent. Sticking the latent manifold, they achieve few-shot single-view reconstruction through latent vector optimization.

Instead, we propose to address multi-view consistency by modeling reconstructions in a dual domain which jointly determines hit/miss classification, where multiple predictions are geometrically grounded (addressing surface discontinuities), and whose quantities are stable w.r.t. changes in incident viewing direction. This domain is the *medial axis*.

The medial axis. The Medial Axis Transform MAT($\mathcal{O}$) is a complete descriptor of shape $\mathcal{O} \subset \mathbb{R}^3$. The MAT is a set of 3D points and radii which together form *medial atoms* (spheres) that are *maximally inscribed* in $\mathcal{O}$. The MAT is invertible, since reconstructing $\mathcal{O}$ from MAT($\mathcal{O}$) amounts to taking the union of the medial atoms.

The MAT has various downstream uses, including 3D shape retrieval [73,74], segmentation [75], and manipulation [76]. MAT-inspired sphere representations have further applications in constructing simplified static [77] and dynamic [78] shapes, closest point computations [79], and volumetric physics simulation [80].

Classically, Bouix et al. [81] compute the MAT from voxel models, Du et al. [76] and Tam et al. [82] compute the MAT from surface meshes, and Rebain et al. [83] iteratively approximate the MAT from oriented point clouds by phrasing the inscription and maximality constraint as optimization energies. The MAT is unstable under noise [83,84], but Tam et al. [82] show how one may prune medial axis branches while maintaining the salient features of the shape.

In the neural literature, Yang et al. [85] predict a set of medial atoms given a sparse surface point cloud, showing how data-driven approaches fare better on sparse and noisy data thanks to its learned priors. Rebain et al. [9] learn a relaxed MAT as a neural field. They model a $\mathbb{R}^3 \rightarrow \mathbb{R}$ *medial field* mapping spatial coordinates to the radius of the medial atom tangential the nearest surface.

There are many ways to define and apply the MAT, and we encourage the reader to view excellent overview by Tagliasacchi et al. [86]. In short there are four MAT definitions: (1) the set of maximally inscribed balls tangent to the surface, (2) the ridges of the (signed) distance $d^{\pm}_{\partial\mathcal{O}}$ (i.e. the grassfire transform), (3) the Maxwell set, i.e. the set of points with more than one nearest surface neighbor, (4) all local axes of reflectional symmetry, i.e. the set of all bi-tangent spheres. In this work we use the first definition.

3. Method

We start by establishing a mathematical framework for oriented ray intersection fields in Section 3.1. We then define our proposed Medial Atom Ray Field (MARF) in Section 3.2, followed by a discussion on how MARFs address the challenges in learning ray fields. We define our neural architecture in Section 3.3, and outline training data pre-processing, losses and optimization strategy in Section 3.4.

3.1. Oriented ray intersection fields

Consider a closed 3D shape $\mathcal{O} \subset \mathbb{R}^3$ with regular surface boundary $\partial\mathcal{O}$, and the oriented ray as the line $\ell \in \mathcal{R}$ with origin $\mathbf{o}$ and unit direction $\hat{\mathbf{q}}$:

$$\ell(t) = \mathbf{o} + t\hat{\mathbf{q}} \tag{2}$$

The *oriented ray intersection field* $f_{\mathcal{O}} : \mathcal{R} \rightarrow \mathbb{R}^3$ maps 3D oriented rays to their nearest intersection points on the surface $\partial\mathcal{O}$. $f_{\mathcal{O}}$ is in essence a single-ray ray caster. Formally $f_{\mathcal{O}}$ maps ℓ to the point $\mathbf{p}_\ell \in \partial\mathcal{O}$ along $\ell(t)$ minimizing t:

$$f_{\mathcal{O}}(\ell) = \mathbf{p}_\ell = \ell\left(\operatorname*{arg\,min}_{t\ :\ \ell(t)\in\partial\mathcal{O}} t\right) \tag{3}$$

$f_{\mathcal{O}}$ is a partial map, since not all rays intersect with the shape. In such cases we may still observe by how much a ray misses, dubbed the *silhouette distance* s_ℓ, which exhibits the property $\nexists\mathbf{p}_\ell \Leftrightarrow s_\ell > 0$:

$$s_\ell = \min_{t\in\mathbb{R},\ \mathbf{x}\in\partial\mathcal{O}} \|\ell(t) - \mathbf{x}\| \tag{4}$$

Differential geometry in rays fields. A *surface normal* $\hat{\mathbf{n}}_\ell$ is a unit vector (in the 2-sphere S^2) whose orientation is orthogonal to the plane tangent at point $\mathbf{p}_\ell$ on the surface $\partial\mathcal{O}$, and whose direction determines the shape exterior ($\mathbb{R}^3 \setminus \mathcal{O}$). Computing normals is not straight forward given the view-dependent ray parametrization $\ell(t) = \mathbf{o} + t\hat{\mathbf{q}}$. We first compute, for each coordinate axis $\hat{\mathbf{e}}_i$, a surface tangent vector as the partial derivative $\mathbf{t}_i = \partial\mathbf{p}_\ell/\partial o_i$, visualized in Fig. 3(b). The normal $\hat{\mathbf{n}}_\ell$, orthogonal to the tangent space, is determined by the cross product of two tangents. But our tangents may, depending on view direction, become zero (since $\hat{\mathbf{q}} \parallel \hat{\mathbf{e}}_i \Rightarrow \mathbf{t}_i = \mathbf{0}$) or change the cross-product handedness determining the exterior. We thus compute the cross product of all three tangent pairs, then modulate their sign and contribution using the viewing direction before summation and normalization:

$$\begin{aligned}\hat{\mathbf{n}}_\ell = \frac{\mathbf{n}'_\ell}{\|\mathbf{n}'_\ell\|},\ \mathbf{n}'_\ell = & -\hat{q}_1\,(\mathbf{t}_2\times\mathbf{t}_3) & = & -\hat{q}_1\left(\frac{\partial\mathbf{p}_\ell}{\partial o_2}\times\frac{\partial\mathbf{p}_\ell}{\partial o_3}\right)\\ & -\hat{q}_2\,(\mathbf{t}_3\times\mathbf{t}_1) & & -\hat{q}_2\left(\frac{\partial\mathbf{p}_\ell}{\partial o_3}\times\frac{\partial\mathbf{p}_\ell}{\partial o_1}\right)\\ & -\hat{q}_3\,(\mathbf{t}_1\times\mathbf{t}_2) & & -\hat{q}_3\left(\frac{\partial\mathbf{p}_\ell}{\partial o_1}\times\frac{\partial\mathbf{p}_\ell}{\partial o_2}\right)\end{aligned} \tag{5}$$

where o_i and $\hat{q}_i$ are the ith scalar components of $\mathbf{o}$ and $\hat{\mathbf{q}}$ from Eq. (2).

Curvature describes how a surface deviates from the tangent plane and is intrinsic to the shape. The curvature κ along a single direction is the reciprocal of the radius of an osculating circle, where positive curves osculate inside. Curvature on 3D surfaces (i.e. 2-manifolds) may be expressed as two *principal* curvatures κ_1, κ_2, respectively the maximum and minimum curvatures. The principal directions of curvature are always perpendicular, except at umbilical points and on flat surfaces where $\kappa_1 = \kappa_2$.

Curvature is contained in the *shape operator* $\mathcal{D}\hat{\mathbf{n}}_\ell$, defined as the total derivative of the unit normal $\hat{\mathbf{n}}_\ell$ along the tangent space [87]. The ray origin $\mathbf{o}$, our input, is not restricted to the tangent space. As such we compute $\mathcal{D}\hat{\mathbf{n}}$ by projecting the total derivative onto the tangent plane:

$$\mathcal{D}\hat{\mathbf{n}}_\ell = \left(\mathbf{I} - \hat{\mathbf{n}}_\ell\hat{\mathbf{n}}_\ell^\top\right)\nabla_{\mathbf{o}}\hat{\mathbf{n}}_\ell \tag{6}$$

The principal curvatures and directions equal the maximal and minimal eigenvalues of the shape operator $\mathcal{D}\hat{\mathbf{n}}_\ell$ and associated eigenvectors. In our case the total derivative $\nabla_{\mathbf{o}}\hat{\mathbf{n}}_\ell$ is a 3×3 matrix with three eigenvectors, but the eigenvector with the smallest absolute eigenvalue is associated with the normal and can be discarded [20]. The mean curvature is half the trace of the shape operator $\mathcal{D}\hat{\mathbf{n}}_\ell$, and the Gaussian curvature is its determinant [19].

We can compute these differentials analytically for continuous neural representations, unlike for meshes which do not admit a continuous normal field in turn requiring an approximation like the discrete shape operator [88]. Computing curvature requires a sufficiently smooth activation function [17,20] since piecewise linear activations have no second derivative.

3.2. The Medial Atom Ray Field (MARF)

We propose learning *Medial Atom Ray Fields* (MARFs), a dual field that also represents the ray intersection field. A MARF $\mathcal{M}_{\mathcal{O}}$ maps an oriented ray $\ell \in \mathcal{R}$ to a *medial atom* (sphere) with center $\mathbf{c}_\ell \in \mathbb{R}^3$ and radius $r_\ell \in \mathbb{R}^+$, such that the atom:

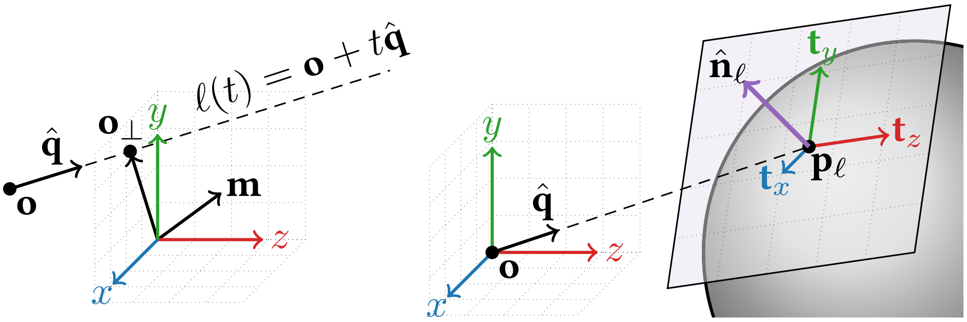
- *intersect* ℓ at the same point ($\mathbf{p}_\ell$) where ℓ intersects the surface $\partial\mathcal{O}$ of $\mathcal{O}$,
- is *tangential* to surface $\partial\mathcal{O}$ at $\mathbf{p}_\ell$ (i.e. share $\hat{\mathbf{n}}_\ell$ from Eq. (5)),
- is fully *inscribed* in shape $\mathcal{O}$, and
- is *maximal*.

The atoms of MARF are thus members of the Medial Axis Transform (MAT) [86] of $\mathcal{O}$. The MARF $\mathcal{M}_{\mathcal{O}}$ relates to the "ray-caster" $f_{\mathcal{O}}$ and its normal $\hat{\mathbf{n}}_\ell$ (Eqs. (3), (5)) as follows:

$$\mathcal{M}_{\mathcal{O}}(\ell) = (\mathbf{c}_\ell, r_\ell)\ :\ \|f_{\mathcal{O}}(\ell) - \mathbf{c}_\ell\| = r_\ell,\ \hat{\mathbf{n}}_\ell = \frac{f_{\mathcal{O}}(\ell) - \mathbf{c}_\ell}{\|f_{\mathcal{O}}(\ell) - \mathbf{c}_\ell\|} \tag{7}$$

To determine the point $\mathbf{p}_\ell$ where the ray ℓ intersects a medial atom $(\mathbf{c}_\ell, r_\ell)$ we solve the system $\mathbf{p}_\ell = \ell(t),\ \|\ell(t) - \mathbf{c}_\ell\| = r_\ell$:

$$\begin{aligned}&\mathbf{p}_\ell = \mathbf{o} + \hat{\mathbf{q}}\left(-(\hat{\mathbf{q}}\cdot(\mathbf{o}-\mathbf{c}_\ell)) \pm \sqrt{\delta_\ell}\right)\\ &\text{where}\quad \delta_\ell = (\hat{\mathbf{q}}\cdot(\mathbf{o}-\mathbf{c}_\ell))^2 - (\|\mathbf{o}-\mathbf{c}_\ell\|^2 - r_\ell^2)\end{aligned} \tag{8}$$



(A) Moment $\mathbf{m}$ and foot $\mathbf{o}_\perp$

(B) Normal derived from surface tangents.

Fig. 3. (A) illustrates the relation between ray origin $\mathbf{o}$ and direction $\hat{\mathbf{q}}$ to the moment $\mathbf{m}$ and perpendicular foot $\mathbf{o}_\perp$ which are invariant to changes in the length of $\mathbf{q}$ and translation of $\mathbf{o}$ along $\mathbf{q}$. (B) illustrates how Eq. (5) determines the normal vector $\hat{\mathbf{n}}_\ell$, orthogonal to the tangent space, by projecting the x, y, z coordinate unit vectors onto the plane tangent at $\mathbf{p}_\ell$ where ray ℓ intersects.

This phrasing of intersection $\mathbf{p}_\ell$ yields up to two real solutions (a near and far hit) when ℓ hits ($\delta_\ell \geq 0$), and a complex solution when ℓ misses ($\delta_\ell < 0$). We can as such use δ_ℓ to determine ray hit/miss classification.

When ray ℓ misses, the real component of $\mathbf{p}_\ell$ equals the orthogonal projection of $\mathbf{c}_\ell$ onto ℓ, yielding the following relation with the silhouette s_ℓ from Eq. (4):

$$s_\ell = \| \operatorname{Re}(\mathbf{p}_\ell) - \mathbf{c}_\ell \| - r_\ell \tag{9}$$

It is cheaper to compute the surface normal $\hat{\mathbf{n}}_\ell$ using the medial atom than to compute the differential in Eq. (5). From here on we use the term “analytical normal” to tell Eq. (5) apart from this “medial normal”:

$$\hat{\mathbf{n}}_\ell = \frac{\mathbf{p}_\ell - \mathbf{c}_\ell}{\|\mathbf{p}_\ell - \mathbf{c}_\ell\|} \tag{10}$$

By construction the medial normal will naturally “roll off” as the ray approaches the edge of the represented shape.

Learning the medial axis. We do not assume the medial axis is available for supervision, meaning the network must discover it during training. Inspired by Rebain et al. [83] we phrase the medial axis conditions – maximality and inscription – as optimization energies. The *maximality* energy induces a positive pressure on radius r_ℓ, increasing the atom size, whereas the *inscription* energy penalizes any medial atom candidate visible from the outside, violating the inscription constraint. We define these losses in Section 3.4. We omit their *pinning* energy, since we pin atoms tangential to the surface hit point $\mathbf{p}_\ell$.

Representing surface discontinuities. Neural fields produce interpolation artifacts near sharp edges and discontinuities due to being Lipschitz continuous on their inputs. We address this by predicting multiple medial atom *candidates* (in this work we predict 16), the winner of which we chose using the following metric:

$$m_{\ell,i} = \begin{cases} \hat{\mathbf{q}} \cdot (\mathbf{p}_{\ell,i} - \mathbf{o}) & \text{if } s_{\ell,i} = 0 \\ \infty & \text{if } s_{\ell,i} > 0 \ \wedge \ \exists k (s_{\ell,k} = 0) \\ s_{\ell,i} & \text{if } \forall k (s_{\ell,k} > 0) \end{cases} \tag{11}$$

The first case is when the ith atom candidate intersects ray ℓ, computing the signed displacement with regard to the ray origin $\mathbf{o}$. The second case is when candidate i misses ℓ but at least one other do hit. The final case is when *all* candidates miss ℓ, in which case the metric falls back on the silhouette distance.

Under this metric we may supervise $\mathbf{p}_{\ell,i}$, $\mathbf{n}_{\ell,i}$ and $s_{\ell,i}$ for candidate $\arg\min_i m_{\ell,i}$. It is not perfect, as discussed in Fig. 4, but it ensures the validity of the aforementioned quantities. From here on if we omit the i subscript, then only the winning atom is concerned.

This metric alone is not enough. Neff et al. [66] discuss how multiple network outputs do not alone improve their results. This is likely because only the winning output receives supervision while the others drift. We observed atoms either going unused, or “fighting” to represent the same geometry. As such we add a “specialization” regularization which incentivizes each candidate to target separate regions. We do so by enforcing a spherical prior distribution, centered in the per-candidate centroid, further detailed in Section 3.4. All atom candidates are subject to the medial axis inscription constraint.

Enforcing multi-view consistency. Ray fields are not multi-view consistent by construction. We observe the following trait of multi-view consistency: surface hit points do not move when the viewing angle changes. For any oriented ray intersection field $f_\mathcal{O}$ (Eq. (3)) this means that if we shift the ray origin $\mathbf{o}$, i.e. its pivot point, to the hit $\mathbf{p}_\ell$, then its derivative w.r.t. view direction must be zero:

$$\exists \mathbf{p}_\ell \Rightarrow \left\| \nabla_{\hat{\mathbf{q}}} f_\mathcal{O}(\mathbf{p}_\ell, \hat{\mathbf{q}}) \right\| = \left\| \nabla_{\hat{\mathbf{q}}} \mathbf{p}_\ell \right\| = 0 \tag{12}$$

This is not a trivial property for signed displacement methods like [67,69] to learn, as they must learn the inverse of the change in displacement origin. But it extends cleanly to fixing the medial atom in place:

$$\exists \mathbf{p}_\ell \Rightarrow \left\| \nabla_{\hat{\mathbf{q}}} \mathcal{M}_\mathcal{O}(\mathbf{p}_\ell, \hat{\mathbf{q}}) \right\| \leq \left\| \nabla_{\hat{\mathbf{q}}} \mathbf{c}_\ell \right\| + \left\| \nabla_{\hat{\mathbf{q}}} r_\ell \right\| = 0 \tag{13}$$

In Section 3.4 we express Eqs. (12), (13) as loss functions.

3.3. Network architecture

We model a neural network Φ_θ with learned parameters θ, optionally conditioned on latent codes, to fit the medial atom ray field $\mathcal{M}_{\mathcal{O}_i}$ from Eq. (7) for shapes $\{\mathcal{O}_i \subset \mathbb{R}^3\}_{i=1}^n$. Shown in Fig. 2 we model the network as a Multi-Layer Perceptron (MLP) with skip connections (inspired by [5,63]) to the middle and final hidden layer. Formally:

$$\begin{aligned} \Phi_\theta(\mathbf{x}) &= \mathbf{W}_k \left(\phi_{k-1} \circ \phi_{k-2} \circ \cdots \circ \phi_0\right) + \mathbf{b}_k \\ \phi_i(\mathbf{x}_i) &= \sigma_i \left(\mathbf{W}_i \mathbf{x}_i + \mathbf{b}_i\right) \\ \theta &= \{(\mathbf{W}_i, \mathbf{b}_i)\}_{i=0}^k \end{aligned} \tag{14}$$

where Φ_θ is the composition of k layers where $\phi_i : \mathbb{R}^{m_i} \rightarrow \mathbb{R}^{n_i}$ is the ith network layer, each applying some affine transformation/linear map on intermediate activation $\mathbf{x}_i$ followed by an element-wise application of an activation σ_i. For all σ_i we use Leaky ReLU and layer normalization [89], but the middle and final σ_i also concatenates the original input $\mathbf{x}_0$, forming skip connections [5]. During training, σ_i also applies dropout.

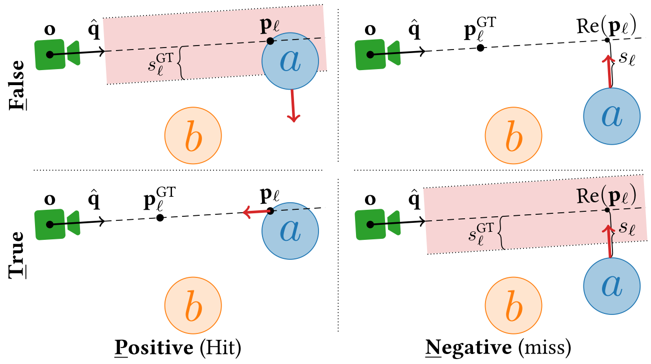


Fig. 4. Four MARF supervision scenarios. For each ray ℓ with origin $\mathbf{o}$ and direction $\hat{\mathbf{q}}$ we predict n atom candidates then pick and supervise the one (a) that minimizes metric m_ℓ (see Eq. (11)). In short the metric favors the atom closest to the ray, then the atom closest to the camera. The **TP** case supervises toward a target intersection point $\mathbf{p}_\ell^{\mathrm{GT}}$, applying a pressure along ℓ (visualized as red arrows), while the **FP**, **FN** and **TN** cases all supervise toward a target silhouette distance s_ℓ^{GT}, shown as a red cylinder lathed about the ray, applying an orthogonal pressure to the atom. Of interest is how atom b might be a better supervision candidate than a in **TP**, **TN** and **FN**, demonstrating a shortcoming of metric m_ℓ.

We encode the ray ℓ as the following 9D input vector with 4 DoF:

$$\mathbf{x} = (\hat{\mathbf{q}}, \mathbf{m}, \mathbf{o}_\perp), \quad \text{where} \quad \begin{aligned} \hat{\mathbf{q}} &= \mathbf{q}/\|\mathbf{q}\| \\ \mathbf{m} &= \mathbf{o} \times \hat{\mathbf{q}} \\ \mathbf{o}_\perp &= \hat{\mathbf{q}} \times \mathbf{m} \end{aligned} \tag{15}$$

where $\mathbf{m}$ is the moment proposed by [63] and $\mathbf{o}_\perp$ is the perpendicular foot used in PRIF [69]. Either $\mathbf{o}_\perp$ or $\mathbf{m}$ would have sufficed: they are of equal length and separated by a 90° rotation, virtually equivalent to the linear maps that neural networks learn. Redundant information however improves learning [2], so we combine them in turn forming an orthogonal basis when ℓ does not pass though the origin. With this in mind we add an extra skip connection to the final layer, since our network predicts points in $\mathbb{R}^3$, improving performance.

The final output is $4 \times n$ features wide, split into n medial atom centers and radii $\{(\mathbf{c}_{\ell,i}, r_{\ell,i})\}_{i=1}^n$, ensuring $r_\ell \in \mathbb{R}^+$ by using the absolute predicted value. Inspired by [13,90] we propose a principled initialization strategy for MARFs. As customary we sample the network parameters θ from a uniform distribution according to [91], but we then scale the final layer weights $\mathbf{W}_k$ by 0.05, in effect reducing the variance of the final predictions. We initialize the final bias $\mathbf{b}_k$ to n random atoms 0.6 units away from the origin and with 0.1 radius. This initialization is multi-view consistent, with each atom candidate starting in a different region as opposed to them all clustering near the origin.

To condition the network on multiple shapes we use the auto-decoder framework by Park et al. [5], where the latent vector $\mathbf{z}_i \in \mathbb{R}^k$ which represents shape $\mathcal{O}_i$ is concatenated with the input coordinates before being fed into the network and are optimized alongside the network weights. We concatenate $\mathbf{z}_i$ at two sites: the initial input and at the middle skip connection. We do not condition the final skip connection.

3.4. Training

We train unconditioned (i.e. single-object) MARFs to represent the Stanford *Armadillo*, *Buddha*, *Bunny*, *Dragon*, and *Lucy* [92], and conditioned (i.e. multi-object) MARFs to represent the *four-legged* object class in COSEG [93].

Data pre-processing. We sample ground truth data points from 3D triangle meshes for training. We scale and translate the meshes to fit inside the unit sphere, then render 200 × 200 depth and normal maps with the rendering pipeline of [94] from 50 equidistant virtual camera views. We unproject the depth maps to 3D points, then compute silhouette distances. We accelerate this with a ball tree index [95] on the hit-points which we sphere-trace along the miss-rays with a 25% step-length. The smallest observed distance approximates the silhouette.

On non-watertight meshes we classify back-face depth pixels as neither hits nor misses, to avoid training on missing data visible as the black holes in Fig. 6(a). We designate these "missing" Ground Truth (GT) pixels as non-hits, i.e. $\nexists \mathbf{p}_\ell^{\mathrm{GT}}$. We still subject these rays to regularization during training – maintaining a fixed batch size in the process – but they otherwise provide no direct supervision. This violates the property $\nexists \mathbf{p}_\ell \Leftrightarrow s_\ell > 0$; we thus introduce the notational convenience in Eq. (16) to "gate" losses where ground truth rays *hit* (h_ℓ^{GT}), *miss* (m_ℓ^{GT}), or are hitting but the intersection data is *missing* ($\bar{h}_\ell^{\mathrm{GT}} \bar{m}_\ell^{\mathrm{GT}}$). Some losses supervise only true hits, denoted $h_\ell h_\ell^{\mathrm{GT}}$.

$$h_\ell^{\mathrm{GT}} = \begin{cases} 1 & \text{if } \exists \mathbf{p}_\ell^{\mathrm{GT}} \\ 0 & \text{if } \nexists \mathbf{p}_\ell^{\mathrm{GT}} \end{cases}, \quad m_\ell^{\mathrm{GT}} = \begin{cases} 1 & \text{if } s_\ell^{\mathrm{GT}} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

Loss. We use the following losses to train our network:

- **Intersection loss** $\mathcal{L}_\mathbf{p}$ and **normal loss** $\mathcal{L}_\mathbf{n}$: When ray ℓ hits we supervise the Euclidean distance between the hit $\mathbf{p}_\ell$ (from Eq. (8)) and ground truth $\mathbf{p}_\ell^{\mathrm{GT}}$. In addition we supervise the cosine similarity between medial normal $\hat{\mathbf{n}}_\ell$ (from Eq. (10)) and ground truth $\hat{\mathbf{n}}_\ell^{\mathrm{GT}}$:

$$\mathcal{L}_\mathbf{p} = \frac{1}{|B|} \sum_{\ell \in B} h_\ell h_\ell^{\mathrm{GT}} \left\| \mathbf{p}_\ell - \mathbf{p}_\ell^{\mathrm{GT}} \right\| \tag{17}$$

$$\mathcal{L}_\mathbf{n} = \frac{1}{|B|} \sum_{\ell \in B} h_\ell h_\ell^{\mathrm{GT}} \frac{\hat{\mathbf{n}}_\ell \cdot \hat{\mathbf{n}}_\ell^{\mathrm{GT}}}{\|\hat{\mathbf{n}}_\ell\| \|\hat{\mathbf{n}}_\ell^{\mathrm{GT}}\|} \tag{18}$$

- **Silhouette loss** $\mathcal{L}_s$ and $\mathcal{L}_h$: We supervise the silhouette distance s_ℓ from Eq. (9) with ground truth s_ℓ^{GT}:

$$\mathcal{L}_s = \frac{1}{|B|} \sum_{\ell \in B} m_\ell^{\mathrm{GT}} \left(s_\ell - s_\ell^{\mathrm{GT}} \right)^2 \tag{19}$$

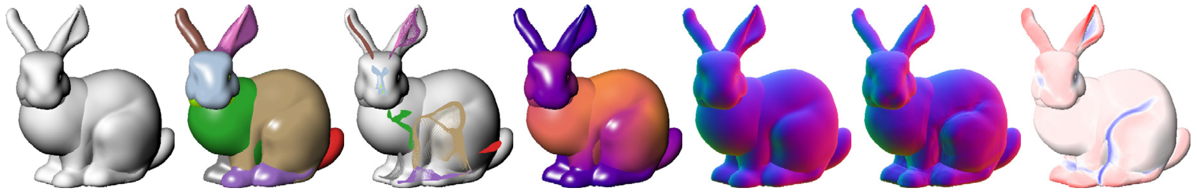
  $\mathcal{L}_s$ only supervise misses, since we found it alone to be insufficient to ensure rays hit when they should. We introduce this additional loss gated on hits, whose strength we tune with a separate hyperparameter:

$$\mathcal{L}_h = \frac{1}{|B|} \sum_{\ell \in B} h_\ell^{\mathrm{GT}} s_\ell^2 \tag{20}$$

- **Maximality regularization** $\mathcal{L}_r$: To ensure the maximality property of medial atoms we apply a constant positive pressure to the radius of all predicted atom candidates, inspired by Rebain et al. [83]:

$$\mathcal{L}_r = \frac{1}{|B| n} \sum_{\ell \in B} \sum_{i=1}^{n} \left| \left(\mathrm{sg}(r_{\ell,i}) + 1 \right) - r_{\ell,i} \right| \tag{21}$$

  where sg(·) returns its input detached from the auto-differentiation graph such that it is considered a constant during back-propagation.
- **Inscription loss** $\mathcal{L}_{ih}$ and $\mathcal{L}_{im}$: To enforce the inscription requirement of medial atoms we supervise all predicted atom candidates w.r.t. a second ray. We randomly permute the order of the training batch of rays B into K, and use K to compute intersections and silhouettes against all n atom



(A) Medial Lambertian (B) Candidate color (C) Medial axis (D) Medial radius (E) Medial normal (F) Analytical normal (G) Medial curvature

Fig. 5. MARF renderings of the Stanford bunny, visualizing the different network outputs. (A) is a Lambertian shading using the normals shown in (E). (B) tints the surface with a unique color associated which atom candidate was chosen by metric $m_{\ell,i}$ in Eq. (11), indicating a learned unsupervised part segmentation. (C) illustrates the medial axis, also known as the topological skeleton, by superimposing the predicted medial atom centers associated with each hitting camera ray, onto (A). (D) maps the predicted radius of the intersected medial atom onto a color scale, in effect visualizing *local thickness* which is useful when approximating translucency. (E) visualizes the normals derived from the intersected medial atoms as RGB (see Eq. (10)), while (F) visualizes the normals derived by differentiating the whole network (see Eq. (5)). (G) visualizes the mean curvature with positive values in red and negative in blue. This curvature is contained in the shape operator (Eq. (6)) which is computed by analytical differentiation of the medial normals shown in (E). (A-E) perform a single network evaluation per pixel, as they are shaded solely using the predicted medial quantities, while (F-G) perform a backward pass to compute the true analytical gradients of the network outputs.

candidates of B. $\mathcal{L}_{ih}$ penalizes atoms that obscure the target intersection of the second ray, while $\mathcal{L}_{im}$ penalizes atoms closer than the second ray silhouette permits.

$$\mathcal{L}_{ih} = \sum_{\substack{\ell_a \in B \\ \ell_b = \rho(\ell_a)}} \sum_{i=1}^{n} h_{\ell_b}^{\mathrm{GT}} h_{\ell_{b|a},i} \frac{\max\left(0,\ \hat{\mathbf{q}}_b \cdot \left(\mathbf{p}_{\ell_b}^{\mathrm{GT}} - \mathbf{p}_{\ell_{b|a},i}\right)\right)}{|B|n} \tag{22}$$

$$\mathcal{L}_{im} = \sum_{\substack{\ell_a \in B \\ \ell_b = \rho(\ell_a)}} \sum_{i=1}^{n} m_{\ell_b}^{\mathrm{GT}} \frac{\max\left(0,\ s_{\ell_b}^{\mathrm{GT}} - s_{\ell_{b|a},i}\right)^2}{|B|n} \tag{23}$$

where $\rho : B \to K$ is a random bijection (a one-to-one mapping) from B to K, and $\mathbf{p}_{\ell_{b|a},i}$ and $s_{\ell_{a|b},i}$ denote the intersection or silhouette of the ith atom candidate predicted with ℓ_a as the network input, but with the ray-atom intersection tests computed using ℓ_b.

- **Specialization regularization** $\mathcal{L}_\sigma$: To avoid atom candidates all clustering on top of each other we introduce $\mathcal{L}_\sigma$, which incentivizes each atom candidate i to cluster its predictions to a smaller volume surrounding a per-candidate centroid $\bar{\mathbf{c}}_i$.

$$\mathcal{L}_\sigma = \frac{1}{n|B|} \sum_{i=1}^{n} \sum_{\ell \in B} \|\mathbf{c}_{\ell,i} - \bar{\mathbf{c}}_i\|^2, \text{ where } \bar{\mathbf{c}}_i = \sum_{\ell \in B} \frac{\mathbf{c}_{\ell,i}}{|B|} \tag{24}$$

This in effect amounts to learning an unsupervised part segmentation of n classes. (Imagine each atom candidate targeting separate limbs of the shape.) This regularization assumes the rays in the training batch cover the whole reconstruction volume.

- **Multi-view loss** $\mathcal{L}_{\mathrm{mv}}$: We phrase the multi-view consistent property in Eqs. (12), (13) as a loss penalizing change in predicted geometry with change in viewing direction. It requires $\mathbf{p}_\ell^{\mathrm{GT}}$ being used as the ray origin, becoming its pivot point. For any oriented ray intersection field, it can be phrased as:

$$\mathcal{L}_{\mathrm{mv}} = \frac{1}{|B|} \sum_{\ell \in B} h_\ell h_\ell^{\mathrm{GT}} \left\| \nabla_{\hat{\mathbf{q}}} \mathbf{p}_\ell \right\|^2 \ :\ \mathbf{o} = \mathbf{p}_\ell^{\mathrm{GT}} \tag{25}$$

For MARFs we use this simplified loss:

$$\mathcal{L}_{\mathrm{mv}} = \frac{1}{|B|} \sum_{\ell \in B} h_\ell h_\ell^{\mathrm{GT}} \left(\left\| \nabla_{\hat{\mathbf{q}}} \mathbf{c}_\ell \right\|^2 + \left\| \nabla_{\hat{\mathbf{q}}} r_\ell \right\|^2 \right) : \mathbf{o} = \mathbf{p}_\ell^{\mathrm{GT}} \tag{26}$$

- **Latent code regularization** $\mathcal{L}_{\mathbf{z}}$: As customary when training auto-decoders we enforce a prior over the latent space to ensure the n embeddings $\{\mathbf{z}_i\}_{i=1}^n$ do not stray too far apart. Like Park et al. [5] we use a spherical prior:

$$\mathcal{L}_{\mathbf{z}} = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{z}_i\|^2 \tag{27}$$

- **Total training loss.** The complete training loss is given by

$$\begin{aligned} \mathcal{L}_{\mathrm{MARF}} = {} & \lambda_{\mathbf{p}} \mathcal{L}_{\mathbf{p}} + \lambda_{\mathbf{n}} \mathcal{L}_{\mathbf{n}} + \lambda_s \mathcal{L}_s + \lambda_h \mathcal{L}_h + \lambda_r \mathcal{L}_r \\ & + \lambda_{ih} \mathcal{L}_{ih} + \lambda_{im} \mathcal{L}_{im} + \lambda_\sigma \mathcal{L}_\sigma + \lambda_{\mathrm{mv}} \mathcal{L}_{\mathrm{mv}} + \lambda_{\mathbf{z}} \mathcal{L}_{\mathbf{z}} \end{aligned} \tag{28}$$

where we tune the λ hyperparameters (see Table 1) to balance the loss terms such that none dominate. We schedule some hyperparameters to ease either in or out during training. The training starts with high specialization loss eased out as the solution becomes more stable. We ease in the normal, multi-view, and latent code losses, as they prove counterproductive early in training.

Table 1
Hyperparameters for Eq. (28); some scheduled using a linear (e_l) or sinusoidal (e_s) easing function (see Eq. (29)) which ease in from 0 to 1 over 'duration' epochs, starting at 'offset' which by default is 0.

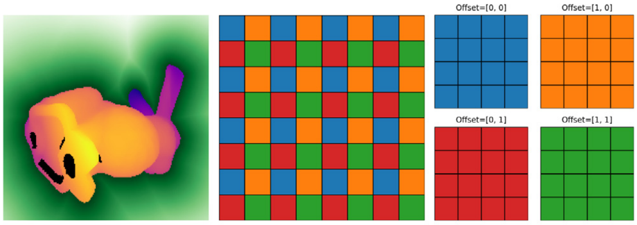
$\lambda_{\mathbf{p}}$	$\lambda_{\mathbf{n}}$	λ_s	λ_h	λ_r	λ_{ih}	λ_{im}	λ_σ	λ_{mv}	$\lambda_{\mathbf{z}}$
2	$\frac{\mathrm{e}_s(85,15)}{4}$	10	100	5×10^{-4}	20	300	$\frac{10-9\,\mathrm{e}_l(40)}{100}$	$\frac{\mathrm{e}_l(50)}{10}$	$0.01^2\,\mathrm{e}_l(30)$

$$\begin{aligned} \mathrm{e}_l(\text{duration}, \text{offset}) &= \operatorname{clamp}\left(\tfrac{\text{epoch}-\text{offset}}{\text{duration}}, 0, 1\right) \\ \mathrm{e}_s(\text{duration}, \text{offset}) &= -\tfrac{1}{2}\left(\cos\left(\pi\ \mathrm{e}_l(\text{duration}, \text{offset})\right) - 1\right) \end{aligned} \tag{29}$$

Optimization. We optimize the network in a stochastic gradient descent scheme, iteratively minimizing the loss in Eq. (28) by tuning the network weights θ through back-propagation. We use the Adam optimizer [96] in PyTorch [97], with default momentum and 5×10^{-6} weight decay, layer normalization [89] and 1% dropout. We warm up to a learning rate of 5×10^{-4} over 100 steps, held for the first 30 epochs, then decay to 1×10^{-4} over the next 170 epochs in a cosine annealing scheme. We train for 200 epochs total, clipping loss gradients exceeding a norm of 1.

4. Experiments

We detail in Section 4.1 our experimental setup. In Section 4.2 evaluation single-shape MARF results, followed by extensive ablation studies in Section 4.3. In Section 4.4 we demonstrate two applications of MARFs in visualization. Finally in Section 4.5 we present a multi-shape MARF, applicable to inverse rendering applications benefiting from learned shape priors.



(A) Typical 200×200 training image (B) Simplified 8×8 image (C) Four 4×4 strided sub-images of (B)

Fig. 6. Splitting single-view depth and silhouette training images (A) into multiple into smaller images during training. Here we show how a simple 8 × 8 training image (B), a representative proxy of (A), is split into 2^2 smaller parts when using a stride size of 2.

4.1. Experimental setup

Training. We model all networks with 8 hidden layers, 512 neurons wide, using PyTorch 1.13 [97] and train with PyTorch Lightning [98] on Python 3.10. To compute analytical derivatives we use the *torch.autograd.grad* function. We reserve 30% of the 50 virtual camera views for validation while tuning hyperparameters, and train for 200 epochs. While the loss in Eq. (28) seemingly assumes a single training image, we construct batches of multiple views and multiple objects to make each batch more diverse. We compute their loss independently and average the results. To fit more views and objects in memory per batch while maintaining a sparse set of training views, we split (as illustrated in Fig. 6) the 200 × 200 training images into 4^2 coarser 50 × 50 sub-images by using a stride of 4. We randomize the order of sub-images across objects and views into batches of 8 in each epoch. We train with CUDA 11.7 using mixed 16 bit float precision and medium matrix multiplication precision. The single-shape MARFs took about 44 min to train on an Nvidia A100 GPU, while the 20-shape MARFs took about 7 h using two A100s, provided by [99].

Evaluation. To render a MARF we evaluate it on the rays associated with each canvas pixel, discard rays that miss, optionally compute analytical network gradient depending on what we visualize, then compute shading. To visualize the medial axis we superimpose the medial atom centers associated with each hitting ray. For quantitative evaluation we sample ground-truth point clouds from each object mesh by casting rays between 4000 viewpoints spaced equidistantly on the enclosing unit-sphere with PyEmbree [100]. In effect this means we evaluate using 4000 camera views while training using only 35. We extract point clouds from the MARF and baseline using the same set of rays, and compute the Precision, Recall and Intersection over Union (IoU) of rays that hit. We then sample 30,000 hit points and compute the Chamfer (CD) and cosine similarity (COS) distance with [101]. Unlike Feng et al. [69] we do not fit a surface to the hit points to compute the CD. The metrics are defined in Appendix.

MARF renders 256 × 256 resolution images at 18.7 frames/s on an Nvidia GTX 3070 8 GB Laptop edition when using medial normals (Eq. (10)). With analytical normals (Eq. (5)) calls we see MARF renders at 4.8 frames/s, while PRIF renders at 5.2 frames/s.

Baseline. We train our reproduction of PRIF by Feng et al. [69] using the same training data, input encoding, learning rate, dropout, normalization and network dimensions as for MARFs. In short, PRIFs predict the signed displacement from the perpendicular foot $\mathbf{o}_\perp$ (Eq. (15)) along the ray direction $\hat{\mathbf{q}}$ – essentially the t in Eq. (2) – as well as *whether* the ray intersects with a second network output supervised with binary cross-entropy loss. We train PRIF with its original loss function denoted $\mathcal{L}_{\text{PRIF}}$. As an experiment we also train PRIF with $2\times$ our normal loss $\mathcal{L}_{\mathbf{n}}$ added (see Eq. (18)), where we compute normals through network differentiation using Eq. (5). We also train PRIF with our multi-view loss $\mathcal{L}_{\text{mv}}$ (see Eq. (25)). Both additions are scaled according to Table 1.

4.2. Learning a single shape

Here we examine MARFs trained from scratch to represent a single shape.

Qualitative results. We visualize in Fig. 7 MARFs and PRIFs trained to represent five Stanford 3D Scanning Repository [92] objects. We visualize the medial quantities represented by MARFs in more detail in Fig. 5 on the bunny. The reconstructions are convincing and stay consistent across views.

MARFs perform well in areas with positive curvature (shaded red in Fig. 5(g)), where atoms stay relatively still w.r.t. a moving ray. In negatively curved areas however the MARFs must learn to "swing" atoms about the curve on the interior, consuming learning capacity. As such MARFs with sufficient number of atom candidates tend to specialize separate atoms to represent each side of sharp negative curves, evident on the body of the bunny in Fig. 9(a).

The MARFs allocated atom candidates where needed. The number of discontinuities possible to represent however is upper bounded by number of atom candidates available. 16 candidates proved insufficient for the dragon in Fig. 7(e), which used a single atom candidate to represent both the upper and lower part of its open mouth. On the other end we find atoms going unused. On the bunny in Fig. 5(b) we only see 9 out of 16 total atom candidates. The other atoms are hidden inside the main body. Fully occluded atom candidates receive no supervision, and if occluded early during training they may not get used at all. We believe this is why the Buddha in Fig. 7(e) fit a single atom to both hands. Maximality regularization is the only pressure counteracting occlusion, but it tends to slide atoms along medial branches when no intersection loss pins it in place [83].

The PRIF baseline reconstructs the training set admirably but struggles with unseen views. MARFs perform better but some pop-ins can be found. In Fig. 7(c) we see a MARF fail to reconstruct the left ear of the bunny. While visible from most camera angles like Fig. 5(a), it is not visible from this one.

MARFs discover sound medial axes on organic shapes like the bunny where the true MAT is simple but struggles on more intricate geometry like the angel Lucy and the Buddha. We shade in Fig. 7(c) using analytically computed normals, which reveal when compared to Fig. 7(d) that MARF in such cases "cheat" by varying the atom radii instead of properly moving the atoms along the medial axis. The lowered medial axis accuracy results in less multi-view consistency, with some warping visible when moving the camera.

The renders in Fig. 7(c) are more accurate than Fig. 7(d), indicating that MARFs prioritize accurate surface intersections over accurate medial normals. We believe this is because the cosine similarity in the normal loss $\mathcal{L}_{\mathbf{n}}$ (Eq. (18)) is in effect a squared distance while the intersection loss $\mathcal{L}_{\mathbf{p}}$ (Eq. (17)) is not. The normal loss proposed in [16] proved unstable however.

Quantitative results. We score MARFs in Table 2 with metrics for both reconstruction quality and ray hit/miss accuracy. There we find that MARF outperform three PRIF [69] variations trained under the same conditions.

In general, MARFs perform better with multi-view loss $\mathcal{L}_{\text{mv}}$ (Eq. (26)) than without, justifying the doubled training time

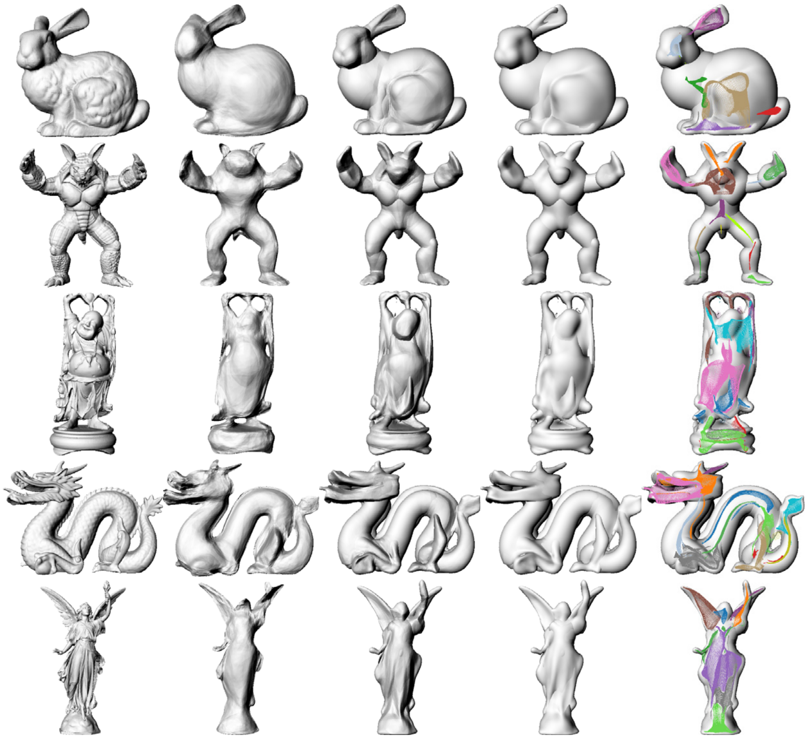

(A) Ground truth (B) ∇PRIF (C) ∇MARF (D) $\mathcal{M}$ MARF (E) $\mathcal{M}$ MARF+axis

Fig. 7. Single-shape MARF and PRIF renderings from a view not present in the training set, with no outlier filtering. ∇ denotes shading with analytical normals (Eq. (5)), and $\mathcal{M}$ denotes shading with medial normals (Eq. (10)).

Table 2
Single-shape results on five Stanford [92] objects shown in Fig. 7. We present with best in **bold** mean CD ($\times 10^4$) and COS scores of reconstruction quality, and IoU scoring ray hit accuracy. We compute IoU on rays cast between 4000 equidistant points, then sample 30,000 hit points to compute CD and COS. We score COS with analytical surface normals (∇) using Eq. (5), For MARF we also score COS with medial normals ($\mathcal{M}$) using Eq. (10). We present MARFs trained with and without multi-view loss $\mathcal{L}_{\text{mv}}$ (Eq. (26)), and PRIFs [69] with its original loss $\mathcal{L}_{\text{PRIF}}$ scored it with and without outlier point filtering. We also train PRIFs with $\mathcal{L}_{\mathbf{n}}$ (Eq. (18)) and with $\mathcal{L}_{\text{mv}}$ (Eq. (25)), scored with filtering. The IoU score considers filtered rays as misses. MARFs are not filtered.

Metrics & objects		PRIF		PRIF	PRIF	MARF + $\mathcal{L}_{\text{mv}}$		MARF − $\mathcal{L}_{\text{mv}}$	
		No Filter	$\mathcal{L}_{\text{PRIF}}$	$+2\mathcal{L}_{\mathbf{n}}$	$+\mathcal{L}_{\text{mv}}$	∇	$\mathcal{M}$	∇	$\mathcal{M}$
CD↓	Armadillo	24.578	22.705	21.653	18.015	2.745		**2.560**	
	Buddha	12.538	12.534	13.991	9.547	2.996		**2.948**	
	Bunny	16.171	15.274	13.746	12.653	**1.816**		2.450	
	Dragon	16.484	16.028	15.180	13.713	**3.187**		4.046	
	Lucy	11.615	10.039	9.841	7.969	**2.064**		2.203	
COS↑	Armadillo	0.597	0.603	0.632	0.644	**0.815**	0.788	0.793	0.762
	Buddha	0.508	0.517	0.503	0.554	**0.715**	0.665	0.706	0.677
	Bunny	0.753	0.757	0.763	0.780	**0.937**	0.924	0.907	0.881
	Dragon	0.558	0.562	0.582	0.583	**0.802**	0.768	0.743	0.698
	Lucy	0.439	0.440	0.462	0.443	**0.630**	0.586	0.624	0.580
IoU↑	Armadillo	84.0%	80.8%	81.5%	81.8%	**92.5%**		91.3%	
	Buddha	91.8%	88.2%	88.9%	90.3%	**93.1%**		91.5%	
	Bunny	93.2%	90.4%	90.9%	91.3%	**95.7%**		94.9%	
	Dragon	88.6%	83.8%	84.7%	86.1%	**92.0%**		90.5%	
	Lucy	86.6%	84.0%	85.0%	85.2%	**89.8%**		88.3%	

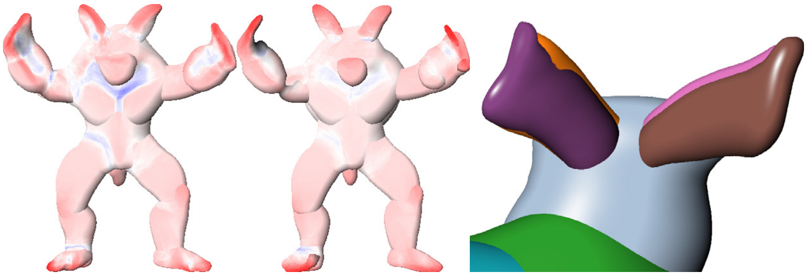
thanks to double back-propagation. The Armadillo is the exception, which as shown in Fig. 7(e) used a single atom candidate to represent both ears. The multi-view loss spikes as the atom quickly "jumps" from one ear to the other. We visualize such a discontinuity in Fig. 9(b).

The benefits of our normal loss $\mathcal{L}_{\mathbf{n}}$ (Eq. (18)) and multi-view loss $\mathcal{L}_{\text{mv}}$ transfers over when applied to PRIF. The exception is the Buddha, which sees a decrease in reconstruction quality.

4.3. Ablations

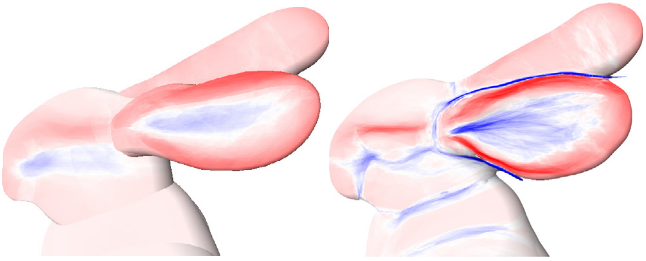
We conduct extensive ablation studies in Table 3 on the terms of our loss function in Eq. (28), as well as on three of our architecture choices: the input encoding scheme, our principled initialization scheme, and number of atom candidates predicted per ray. For each experiment we train five MARFs, one for each of the five Stanford objects explored in Section 4.2.

Our ray input encoding scheme outperforms both LFN [63] and PRIF [69]. While it does not raise the spectral bias nor provide



(A) Trained with $\mathcal{L}_{\mathbf{n}}$ (B) Trained without $\mathcal{L}_{\mathbf{n}}$ (C) Overfitting to approximated training silhouettes

Fig. 8. (A–B) compares two MARFs (shaded with mean curvature) trained with and without surface normal supervision. The latter fails to represent negative curvature (blue) in its medial normals (Eq. (10)). (C), shaded with unique colors per atom candidate, illustrates what happens when using too strong silhouette supervision. The MARF overfits small atom candidates near edges to reconstruct the inaccurate ground truth silhouettes, which we approximated using sphere-tracing.



(A) Good specialization (B) Poor specialization

Fig. 9. Two Stanford bunny MARFs shaded with mean curvature using medial normals. In (A) we compare the MARF from Fig. 5 against a MARF trained without our initialization scheme (Section 3.3) in (B), which failed to specialize its atom candidates to deal with discontinuities. (B) uses a single atom candidate to represent the head and both ears. While negatively curved areas (blue) on the body are better represented, it also produces artifacts where atoms "jump" across discontinuities under a Lipschitz bound.

any additional information it does raise the Lipschitz bound, demonstrating how much a positional encoding [21,64] scheme would benefit neural ray fields.

We see accuracy increase and decrease with the number of atom candidates predicted, diminishing in return as it increases. Past 16 candidates we observe a decline in hit precision, where atoms become prune to occlude each other.

Without our initialization scheme or specialization regularization, we find the training getting stuck in a local minima where the atom candidates fail to target separate limbs, illustrated in Fig. 9, causing interpolation artifacts near discontinuities.

Without intersection loss we find the atoms still intersect the ray thanks to the silhouette loss, but nothing pins the atoms tangential to the ray-surface intersection point. Inscription loss constrains the atoms to stay on the interior, but no counteracting force "pulls" them back out toward the surface. Maximality regularization instead slides the atoms down medial branches where larger medial radii are supported, in effect eroding the represented shape.

Without silhouette loss we observe a large drop in ray hit/miss accuracy. While the intersection and normal losses only supervise true hits, they are still able to gradually "roll" atoms to where they are needed thanks to their non-zero size. If we further increase the silhouette loss we find IoU and CD improve, but the normal accuracy decrease. We suspect our approximate silhouette ground truths are too inaccurate, causing overfitting where some atom candidates are specialized to extend the outline, as evident in Fig. 8(c).

Without normal loss we see ray hit precision and surface reconstruction improve, at the cost of medial normal accuracy and hit recall. In Fig. 8(b) we show a MARF without normal supervision which fails to represent negative curvatures. In theory the combination of intersection and inscription loss should suffice making atoms "swing" about negative curvature, indicating that our inscription testing may be too coarse.

We enforce inscription on each ray using just one other ray in the batch chosen at random. Without inscription loss we find the ray hit recall improving as expected, but at the cost of a lowered ray hit precision. Atom candidates that miss the ray (i.e. the majority) become free to violate the inscription constraint, becoming prone to getting stuck in local minima as a result. Further increasing inscription loss overpowers the other losses, in effect eroding the represented shape.

Without maximality regularization we observe a non-significant decline on all metrics. In our object-centric setup, its function overlaps with the silhouette loss which stretches atoms to fill in the shape contour from all camera views. When we further increase the effect of maximality regularization we find reconstruction quality improving, but also more false ray hits. This indicates that the atoms grow beyond the confines of the surface boundary. While the amount of inscription loss seems ideal, we believe its resolution may be insufficient.

Without multi-view loss the accuracy drops across all metrics. The surface visibly wobbles as we move the camera, indicating that the MARF is overfitting to the sparse set of training views. Adding too much multi-view loss also causes a drop in accuracy, effectively pruning branches of the medial axis, which is a common strategy to simplify shapes [82].

4.4. Applications in visualization

In this section we showcase two real-time applications in visualization made possible due to the medial quantities predicted by MARFs.

Translucency. Light traveling inside translucent objects attenuates and scatters rapidly. How this phenomenon appears on the surface is commonly approximated using some measure of local thickness, for which the medial radius predicted by MARFs, also known as the local feature size, is an excellent candidate. We showcase in Fig. 10 approximate translucency, using the shading model of Barré-Brisebois et al. [102]. It contributes the following shading coefficient at each point $\mathbf{p}_\ell$ on surface $\partial\mathcal{O}$:

$$k_{\text{translucency}} = \frac{1}{r_\ell + \epsilon} \cdot \max\left(\hat{\mathbf{q}} \cdot \left(s\hat{\mathbf{n}}_\ell - \hat{\mathbf{l}}\right), 0\right)^p \tag{30}$$

where r_ℓ is the thickness (medial radius) at $\mathbf{p}_\ell$, $\epsilon = 0.05$ avoids division by zero, $\hat{\mathbf{q}}$ is the unit ray direction (Eq. (2)), $s = 0.08$ is a distortion determining the amount of subsurface scattering, $\mathbf{n}_\ell$ is the medial normal (as per Eq. (10)), $\hat{\mathbf{l}}$ is the incident light unit vector, and $p = 16$ is a sharpness coefficient.

Table 3
Ablation studies. We present reconstruction quality CD and COS scores, and ray hit IoU, Precision. and Recall scores. Each row is the average score of five single-shape MARFs, one for each object explored in Section 4.2. The first row is our proposed configuration, while the following rows make a single modification each. We mark scores 0.5% worse than MARF **red** and scores 0.5% better **green**. ∇ denotes analytical normals (Eq. (5)) and $\mathcal{M}$ denotes medial normals (Eq. (10)).

Configuration		IoU↑	P↑	R↑	CD↓	COS↑	
					$\times 10^4$	∇	$\mathcal{M}$
MARF	Table 1	92.6%	95.1%	97.2%	2.56	0.780	0.746
LFN [63] encoding	$\mathbf{x} = (\hat{\mathbf{q}}, \mathbf{m})$	92.2%	**94.5%**	97.4%	**2.81**	**0.761**	**0.724**
PRIF [69] encoding	$\mathbf{x} = (\hat{\mathbf{q}}, \mathbf{o}_\perp)$	**92.1%**	94.7%	97.1%	**2.70**	**0.771**	**0.737**
No init scheme.	Section 3.3	**91.8%**	94.8%	**96.7%**	**2.79**	**0.763**	**0.724**
1 atom candidate		**87.4%**	95.2%	**91.5%**	**4.54**	**0.720**	**0.679**
4 atom candidates		**90.7%**	95.0%	**95.2%**	**3.36**	**0.761**	**0.722**
8 atom candidates		**91.9%**	95.1%	**96.4%**	**2.67**	**0.770**	**0.736**
32 atom candidates		92.7%	94.7%	97.7%	**2.60**	0.778	0.749
64 atom candidates		92.7%	**94.6%**	**97.8%**	**2.39**	0.778	0.747
No intersection loss	$0\lambda_{\mathbf{p}}$	**91.3%**	**93.9%**	97.0%	**12.29**	**0.546**	**0.528**
No silhouette loss	$0\lambda_s$ $0\lambda_h$	**87.6%**	**90.5%**	**96.5%**	**3.69**	**0.744**	**0.709**
More silhouette loss	$5\lambda_s$ $5\lambda_h$	**93.4%**	**96.2%**	97.0%	**2.45**	**0.772**	**0.738**
No normal loss	$0\lambda_{\mathbf{n}}$	92.7%	**96.2%**	**96.3%**	**2.15**	0.782	**0.725**
No inscription loss	$0\lambda_{ih}$ $0\lambda_{im}$	92.3%	**93.7%**	**98.4%**	**2.67**	0.778	0.747
More inscription loss	$5\lambda_{ih}$ $5\lambda_{im}$	92.3%	95.5%	**96.5%**	**2.62**	0.776	**0.742**
No maximality reg.	$0\lambda_r$	92.4%	95.0%	97.1%	2.56	0.776	0.744
More maximality reg.	$100\lambda_r$	92.5%	**94.5%**	**97.8%**	**2.54**	0.781	0.748
No specialization reg.	$0\lambda_\sigma$	92.5%	95.0%	97.3%	**2.58**	**0.776**	0.743
No multi-view loss	$0\lambda_{\text{mv}}$	**91.3%**	**94.6%**	**96.3%**	**2.84**	**0.755**	**0.720**
More multi-view loss	$2\lambda_{\text{mv}}$	92.3%	94.8%	97.2%	**2.66**	**0.769**	**0.734**

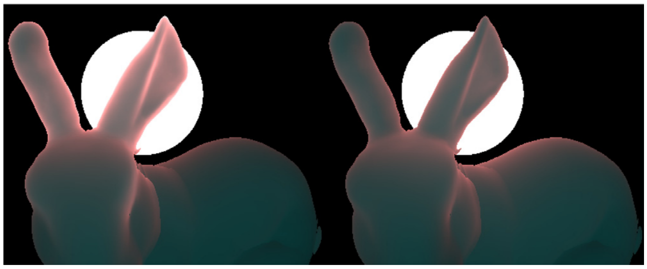

(A) Medial radius thickness (B) Uniform thickness

Fig. 10. Approximate translucency and subsurface scattering. In (A) we use the medial radius (shown in Fig. 5(d)) as a measure of local thickness, while (B) assumes uniform thickness, here set to the mean medial radius for a fair comparison. We render each pixel independently using only a single network evaluation and no differentiation.

Anisotrophy. To show how we can compute the full shape operator $\mathcal{D}\hat{\mathbf{n}}_\ell$ (from Eq. (6)) of a MARF using only a *single* network differentiation, we shade in Fig. 11 a MARF with the anisotropic specular reflectance model of Ward [103]. Anisotropic materials feature view-dependent properties, in our case reflectance, whose distribution Ward determines using two perpendicular surface tangents. For these a good fit are the principal directions of curvature $\hat{\mathbf{v}}_1$ and $\hat{\mathbf{v}}_2$, the eigenvectors of $\mathcal{D}\hat{\mathbf{n}}_\ell$. The Ward model contributes the following specular coefficient at each point $\mathbf{p}_\ell$ on surface ∂O:

$$k_{\text{specular}} = \frac{\exp\left(-2\dfrac{\left(\frac{\hat{\mathbf{h}}\cdot\hat{\mathbf{v}}_1}{a_1}\right)^2 + \left(\frac{\hat{\mathbf{h}}\cdot\hat{\mathbf{v}}_2}{a_2}\right)^2}{1 + \hat{\mathbf{n}}_\ell \cdot \hat{\mathbf{h}}}\right)}{4\pi a_1 a_2 \sqrt{(\hat{\mathbf{n}}_\ell \cdot \hat{\mathbf{l}})(\hat{\mathbf{n}}_\ell \cdot \hat{\mathbf{q}})}} \tag{31}$$

where $\hat{\mathbf{n}}_\ell$ is the medial normal from Eq. (10), $\hat{\mathbf{l}}$ is the incident light unit vector, $\hat{\mathbf{h}} = (\hat{\mathbf{l}} + \hat{\mathbf{q}})/\|\hat{\mathbf{l}} + \hat{\mathbf{q}}\|$, and a_1 and a_2 are the standard deviations of anisotropy along principal directions of curvature $\hat{\mathbf{v}}_1$ and $\hat{\mathbf{v}}_2$.

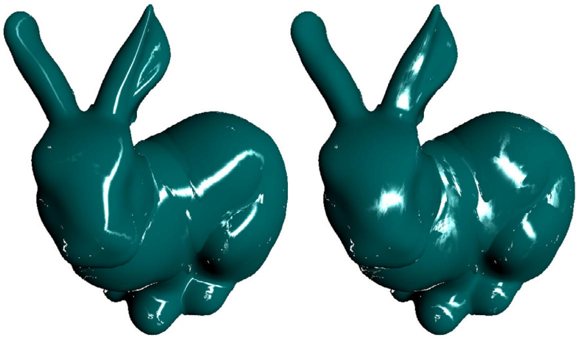

(A) $a_1{=}0.05,\ \ a_2{=}0.3$ (B) $a_1{=}0.3,\ \ a_2{=}0.05$

Fig. 11. A Stanford bunny MARF shaded with Ward anisotropic specular reflectance [103] determined by principal directions of curvature. We render each pixel independently with a single forward and backward pass. a_1 and a_2 (see Eq. (31)) determine the anisotropic deviation along each principal direction.

4.5. Learning multiple shapes

Here we examine a MARF trained to represent multiple shapes. We visualize in Fig. 12 MARF reconstructions of the *four-legged* COSEG [93] object class, reconstructed from learned auto-decoding latent vectors in $\mathbb{R}^{16}$. We also show, to demonstrate how smooth the latent space is, in-between interpolations in latent space. This MARF has a CD score of 2.424×10^{-4}, an analytical COS score of 0.868, and a medial COS score of 0.843, and a 90.9% IoU score.

MARFs proves able to represent a space of multiple species with different articulations with a consistent part segmentation. The latent space appears smooth despite a sparse training set, with meaningful interpolations. On some in-betweens the predicted atoms fail to intersect the ray, visible on the legs leg of the giraffe-dromedary interpolation and on the dogs. This should improve with more training shapes.

5. Conclusion

The novel 3D object representation MARF is a neural ray-to-surface mapping that outperforms prior work, achieving accurate

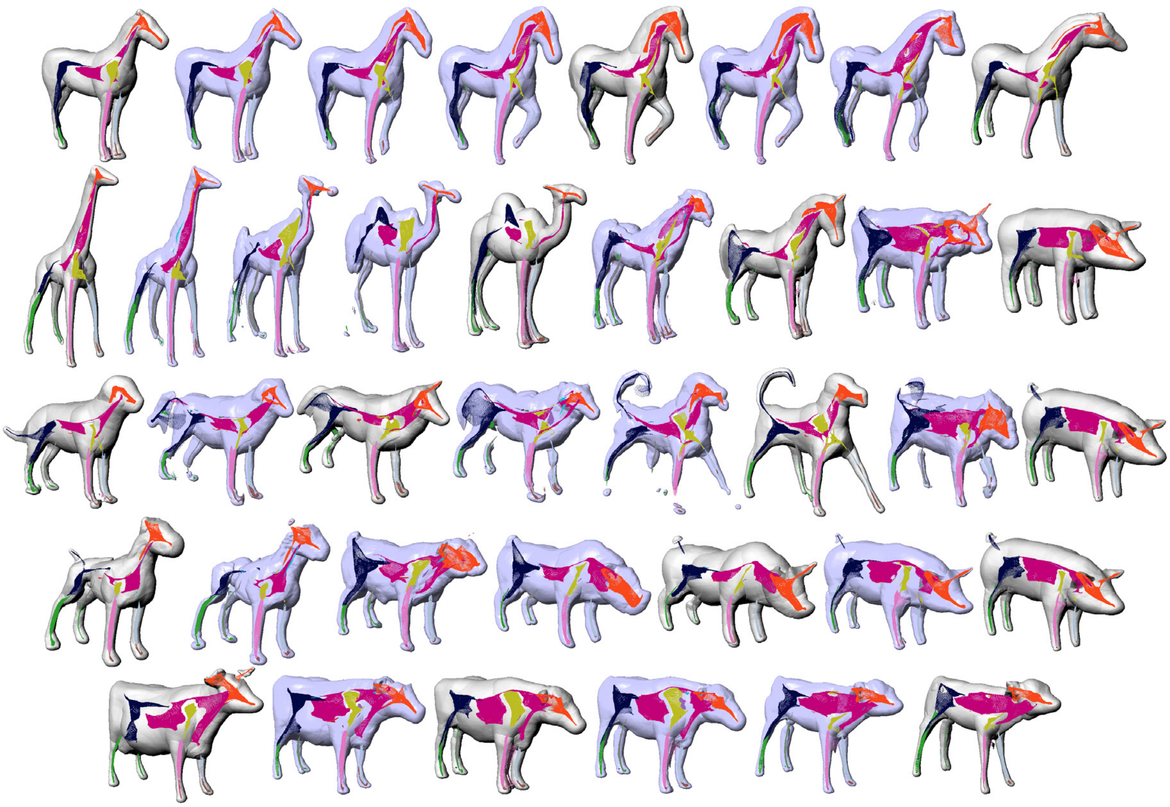


Fig. 12. Row-wise MARF interpolations in latent space. We illustrate the medial axis by superimposing the medial atom centers associated with each hitting camera ray on top of a Lambertian shading using analytical normals (Eq. (5)). We trained this MARF on the COSEG [93] "four-legged" object class, featuring a total of 20 shapes. Gray renders use known auto-decoder latent vectors, while the blue-tinted renders are in-betweens interpolations. Despite the sparse dataset, we find the MARF propose smooth and meaningful in-betweens.

surface rendering with a single network evaluation per camera ray. The geometrically grounded medial representation of MARFs offers more insight while benefiting reconstruction quality, multi-view consistency, and representing discontinuities. We demonstrated how its medial quantities can be used in visualization and inform part-based segmentation. While learning ray-fields remains a difficult problem, we find our results exciting, warranting further study.

Limitations. Like prior neural ray fields, MARFs assume the camera ray is cast from infinitely far away. This makes rendering views where the camera is placed in-between occluders, such as overhangs, impossible. While this does not affect inter-object scatter rays if adapted to a global-illumination ray-tracing setup, it will affect intra-object bounces whose contribution to illumination must be learned/baked.

Future work. There are many challenges to address concerning both MARFs and neural ray fields in general. Ray fields lack an analog to both positional encoding and local conditioning common in Cartesian neural fields, which drastically improve their fidelity. Our proposed multi-view loss requires 3D supervision, in turn requiring two forward passes if adapted to 2D data. For MARFs in particular, we look to explore less naive candidate selection strategies that select the atom candidate best suited to receive supervision, which is not necessarily the one closest to the ray. We would further like to explore alternatives to fully opaque atoms such that no atom are fully occluded from supervision. Finally, work is needed to reduce the number of MARF loss terms, reducing the effort required to balance their contribution.

CRediT authorship contribution statement

Peder Bergebakken Sundt: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Software, Validation, Visualization, Writing – original draft. **Theoharis Theoharis:** Writing – review & editing, Supervision, Resources, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Publicly available datasets were used. Our preprocessing code is provided in the code repository along with downloads for precomputed data and trained network weights.

Acknowledgments

The authors would like to thank the IDUN cluster [99] at NTNU for computing resources.

Appendix

Precision, Recall, and Intersection over Union (IoU). IoU quantifies the overlap between two binary classifiers, in our case ray hit/miss classification. Precision and recall scores the relevance of the classification. For a batch of rays B with hits being positive, the Precision, Recall, and IoU is:

$$\begin{aligned}
\text{Precision} &= \frac{TP}{TP+FP} &&= \frac{\left|\left\{\ell \in B : s_\ell = 0 \wedge s_\ell^{\mathrm{GT}} = 0\right\}\right|}{\left|\left\{\ell \in B : s_\ell = 0\right\}\right|} \\
\text{Recall} &= \frac{TP}{TP+FN} &&= \frac{\left|\left\{\ell \in B : s_\ell = 0 \wedge s_\ell^{\mathrm{GT}} = 0\right\}\right|}{\left|\left\{\ell \in B : s_\ell^{\mathrm{GT}} = 0\right\}\right|} \\
\text{IoU} &= \tfrac{TP}{TP+FP+FN} &&= \frac{\left|\left\{\ell \in B : s_\ell = 0 \wedge s_\ell^{\mathrm{GT}} = 0\right\}\right|}{\left|\left\{\ell \in B : s_\ell = 0 \vee s_\ell^{\mathrm{GT}} = 0\right\}\right|}
\end{aligned} \tag{32}$$

Chamfer Distance (CD) and Cosine Similarity (COS). CD is the "average-case" distance between two point clouds U and V. COS scores orientation using the same matching between U and V as CD, and computes the normal vector cosine similarity.

$$\begin{aligned}
\text{CD} &= \frac{1}{|U|}\sum_{\mathbf{u}\in U}\min_{\mathbf{v}\in V}\|\mathbf{u}-\mathbf{v}\| \\
&+ \frac{1}{|V|}\sum_{\mathbf{v}\in V}\min_{\mathbf{u}\in U}\|\mathbf{u}-\mathbf{v}\| \\
\text{COS} &= \frac{1}{|U|}\sum_{\mathbf{u}\in U}\hat{\mathbf{n}}_u \cdot \hat{\mathbf{n}}_{\arg\min_{\mathbf{v}\in V}\|\mathbf{u}-\mathbf{v}\|} \\
&+ \frac{1}{|V|}\sum_{\mathbf{v}\in V}\hat{\mathbf{n}}_v \cdot \hat{\mathbf{n}}_{\arg\min_{\mathbf{u}\in U}\|\mathbf{u}-\mathbf{v}\|}
\end{aligned} \tag{33}$$

where $\hat{\mathbf{n}}_{\mathbf{x}}$ is the unit normal vector of oriented point $\mathbf{x}$.

References

[1] Bartlett PL, Foster DJ, Telgarsky M. Spectrally-normalized margin bounds for neural networks. In: Proceedings of the 31st international conference on neural information processing systems. Red Hook, NY, USA: Curran Associates Inc.; 2017, p. 6241–50.

[2] Rahaman N, Baratin A, Arpit D, Draxler F, Lin M, Hamprecht F, et al. On the spectral bias of neural networks. In: Chaudhuri K, Salakhutdinov R, editors. Proceedings of the 36th international conference on machine learning, Vol. 97. PMLR; 2019, p. 5301–10.

[3] Mescheder L, Oechsle M, Niemeyer M, Nowozin S, Geiger A. Occupancy networks: Learning 3D reconstruction in function space. In: 2019 IEEE/CVF conference on computer vision and pattern recognition. CVPR, Long Beach, CA, USA: IEEE; 2019, p. 4455–65. http://dx.doi.org/10.1109/CVPR.2019.00459.

[4] Chen Z, Zhang H. Learning implicit fields for generative shape modeling. 2019.

[5] Park JJ, Florence P, Straub J, Newcombe R, Lovegrove S. Deepsdf: Learning continuous signed distance functions for shape representation. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR). Long Beach, CA, USA: IEEE; 2019, p. 165–74. http://dx.doi.org/10.1109/CVPR.2019.00025.

[6] Chibane J, Mir A, Pons-Moll G. Neural unsigned distance fields for implicit function learning. In: Advances in neural information processing systems (NeurIPS). 2020.

[7] Venkatesh R, Karmali T, Sharma S, Ghosh A, Babu RV, Jeni LA, et al. Deep implicit surface point prediction networks. In: Proceedings of the IEEE/CVF international conference on computer vision. 2021, p. 12653–62.

[8] Chi C, Song S. GarmentNets: Category-level pose estimation for garments via canonical space shape completion. In: The IEEE international conference on computer vision (ICCV). 2021, p. 10.

[9] Rebain D, Li K, Sitzmann V, Yazdani S, Yi KM, Tagliasacchi A. Deep medial fields. 2021.

[10] Simeonov A, Du Y, Tagliasacchi A, Tenenbaum JB, Rodriguez A, Agrawal P, et al. Neural descriptor fields: SE(3)-equivariant object representations for manipulation. 2021.

[11] Frankle J, Carbin M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In: International conference on learning representations. 2019.

[12] Xie Y, Takikawa T, Saito S, Litany O, Yan S, Khan N, et al. Neural fields in visual computing and beyond. 2021.

[13] Atzmon M, Lipman Y. SAL: Sign agnostic learning of shapes from raw data. 2020.

[14] Baydin AG, Pearlmutter BA, Radul AA, Siskind JM. Automatic differentiation in machine learning: A survey. J Mach Learn Res 2018;18:1–43.

[15] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 2019;378:686–707. http://dx.doi.org/10.1016/j.jcp.2018.10.045.

[16] Gropp A, Yariv L, Haim N, Atzmon M, Lipman Y. Implicit geometric regularization for learning shapes. 2020.

[17] Sitzmann V, Martel JNP, Bergman AW, Lindell DB, Wetzstein G. Implicit neural representations with periodic activation functions. In: Proc. NeurIPS. 2020.

[18] Lindell DB, Martel JNP, Wetzstein G. AutoInt: Automatic integration for fast neural volume rendering. In: Proceedings of the conference on computer vision and pattern recognition (CVPR). Nashville, TN, USA: IEEE; 2021, p. 14551–60. http://dx.doi.org/10.1109/CVPR46437.2021.01432.

[19] Yang G, Belongie S, Hariharan B, Koltun V. Geometry processing with neural fields. In: Advances in neural information processing systems, Vol. 34. Curran Associates, Inc.; 2021, p. 22483–97.

[20] Novello T, Schardong G, Schirmer L, da Silva V, Lopes H, Velho L. Exploring differential geometry in neural implicits. Comput Graph 2022;108:49–60. http://dx.doi.org/10.1016/j.cag.2022.09.003.

[21] Mildenhall B, Srinivasan PP, Tancik M, Barron JT, Ramamoorthi R, Ng R. NeRF: Representing scenes as neural radiance fields for view synthesis. 2020.

[22] Tewari A, Fried O, Thies J, Sitzmann V, Lombardi S, Sunkavalli K, et al. State of the art on neural rendering. 2020.

[23] Tewari A, Thies J, Mildenhall B, Srinivasan P, Tretschk E, Wang Y, et al. Advances in neural rendering. 2021.

[24] Mildenhall B, Hedman P, Martin-Brualla R, Srinivasan PP, Barron JT. NeRF in the dark: High dynamic range view synthesis from noisy raw images. In: 2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR). New Orleans, LA, USA: IEEE; 2022, p. 16169–78. http://dx.doi.org/10.1109/CVPR52688.2022.01571.

[25] Mildenhall B, Srinivasan PP, Ortiz-Cayon R, Kalantari NK, Ramamoorthi R, Ng R, et al. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Trans Graph 2019;38:1–4. http://dx.doi.org/10.1145/3306346.3322980.

[26] Baatz H, Granskog J, Papas M, Rousselle F, Novák J. NeRF-tex: Neural reflectance field textures. In: Eurographics symposium on rendering. The Eurographics Association; 2021, p. 13.

[27] Goli L, Rebain D, Sabour S, Garg A, Tagliasacchi A. Nerf2nerf: Pairwise registration of neural radiance fields. 2022, http://dx.doi.org/10.48550/arXiv.2211.01600.

[28] Guo M, Fathi A, Wu J, Funkhouser T. Object-centric neural scene rendering. 2020.

[29] Park K, Sinha U, Hedman P, Barron JT, Bouaziz S, Goldman DB, et al. HyperNeRF: A higher-dimensional representation for topologically varying neural radiance fields. 2021.

[30] Chen J, Zhang Y, Kang D, Zhe X, Bao L, Jia X, et al. Animatable neural radiance fields from monocular RGB videos. 2021.

[31] Peng S, Dong J, Wang Q, Zhang S, Shuai Q, Zhou X, et al. Animatable neural radiance fields for modeling dynamic human bodies. In: ICCV. 2021.

[32] Tschernezki V, Larlus D, Vedaldi A. NeuralDiff: Segmenting 3D objects that move in egocentric videos. In: Proceedings of the international conference on 3D vision (3DV). 2021.

[33] Max N. Optical models for direct volume rendering. IEEE Trans Vis Comput Graphics 1995;1:99–108. http://dx.doi.org/10.1109/2945.468400.

[34] Hart JC. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. Vis Comput 1996;12:527–45. http://dx.doi.org/10.1007/s003710050084.

[35] Knodt J, Baek S-H, Heide F. Neural ray-tracing: Learning surfaces and reflectance for relighting and view synthesis. 2021.

[36] Chibane J, Alldieck T, Pons-Moll G. Implicit functions in feature space for 3D shape reconstruction and completion. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR). Seattle, WA, USA: IEEE; 2020, p. 6968–79. http://dx.doi.org/10.1109/CVPR42600.2020.00700.

[37] Genova K, Cole F, Vlasic D, Sarna A, Freeman WT, Funkhouser T. Learning shape templates with structured implicit functions. 2019.

[38] Jiang CM, Sud A, Makadia A, Huang J, Nießner M, Funkhouser T. Local implicit grid representations for 3D scenes. 2020.

[39] Chabra R, Lenssen JE, Ilg E, Schmidt T, Straub J, Lovegrove S, et al. Deep local shapes: Learning local SDF priors for detailed 3D reconstruction. 2020.

[40] Tretschk E, Tewari A, Golyanik V, Zollhöfer M, Stoll C, Theobalt C. PatchNets: Patch-based generalizable deep implicit 3D shape representations. In: Vedaldi A, Bischof H, Brox T, Frahm J-M, editors. Computer Vision – ECCV 2020, Vol. 12361. Cham: Springer International Publishing; 2020, p. 293–309. http://dx.doi.org/10.1007/978-3-030-58517-4_18.

[41] Reiser C, Peng S, Liao Y, Geiger A. KiloNeRF: Speeding up neural radiance fields with thousands of tiny MLPs. In: International conference on computer vision (ICCV). 2021.

[42] Rebain D, Jiang W, Yazdani S, Li K, Yi KM, Tagliasacchi A. DeRF: Decomposed radiance fields. 2021, p. 14153–61.

[43] Lindell DB, Van Veen D, Park JJ, Wetzstein G. BACON: Band-limited coordinate networks for multiscale scene representation. In: CVPR. 2022.

[44] Martel JNP, Lindell DB, Lin CZ, Chan ER, Monteiro M, Wetzstein G. ACORN: Adaptive coordinate networks for neural representation. ACM Trans Graph (SIGGRAPH) 2021.

[45] Takikawa T, Litalien J, Yin K, Kreis K, Loop C, Nowrouzezahrai D, et al. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. 2021, p. 11358–67.

[46] Peng S, Niemeyer M, Mescheder L, Pollefeys M, Geiger A. Convolutional occupancy networks. In: European conference on computer vision (ECCV). 2020.

[47] Xu Q, Wang W, Ceylan D, Mech R, Neumann U. DISN: Deep implicit surface network for high-quality single-view 3D reconstruction. In: Wallach H, Larochelle H, Beygelzimer A, Alché-Buc Fd', Fox E, Garnett R, editors. Advances in neural information processing systems, Vol. 32. Curran Associates, Inc.; 2019, p. 492–502.

[48] Müller T, Evans A, Schied C, Keller A. Instant neural graphics primitives with a multiresolution hash encoding. 2022, p. 13.

[49] Fridovich-Keil S, Yu A, Tancik M, Chen Q, Recht B, Kanazawa A. Plenoxels: Radiance fields without neural networks. In: 2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR). New Orleans, LA, USA: IEEE; 2022, p. 5491–500. http://dx.doi.org/10.1109/CVPR52688.2022.00542.

[50] Yu A, Li R, Tancik M, Li H, Ng R, Kanazawa A. PlenOctrees for real-time rendering of neural radiance fields. In: ICCV. 2021.

[51] Karnewar A, Ritschel T, Wang O, Mitra N. ReLU fields: The little non-linearity that could. In: Special interest group on computer graphics and interactive techniques conference proceedings. Vancouver BC Canada: ACM; 2022, p. 1–9. http://dx.doi.org/10.1145/3528233.3530707.

[52] Hedman P, Srinivasan PP, Mildenhall B, Barron JT, Debevec P. Baking neural radiance fields for real-time view synthesis. In: 2021 IEEE/CVF international conference on computer vision (ICCV). Montreal, QC, Canada: IEEE; 2021, p. 5855–64. http://dx.doi.org/10.1109/ICCV48922.2021.00582.

[53] Reiser C, Szeliski R, Verbin D, Srinivasan PP, Mildenhall B, Geiger A, et al. MERF: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. 2023, http://arxiv.org/abs/2302.12249 (accessed March 24, 2023).

[54] Yariv L, Gu J, Kasten Y, Lipman Y. Volume rendering of neural implicit surfaces. 2021.

[55] Oechsle M, Peng S, Geiger A. UNISURF:Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In: International conference on computer vision (ICCV). 2021.

[56] Li C, Li S, Zhao Y, Zhu W, Lin Y. RT-NeRF:Real-time on-device neural radiance fields towards immersive AR/VR rendering. In: Proceedings of the 41st IEEE/ACM international conference on computer-aided design. San Diego California: ACM; 2022, p. 1–9. http://dx.doi.org/10.1145/3508352.3549380.

[57] Lin H, Peng S, Xu Z, Yan Y, Shuai Q, Bao H, et al. Efficient neural radiance fields for interactive free-viewpoint video. In: SIGGRAPH Asia 2022 conference papers. Daegu Republic of Korea: ACM; 2022, p. 1–9. http://dx.doi.org/10.1145/3550469.3555376.

[58] Morozov N, Rakitin D, Desheulin O, Vetrov D, Struminsky K. Differentiable rendering with reparameterized volume sampling. 2023, http://dx.doi.org/10.48550/arXiv.2302.10970.

[59] Feng BY, Varshney A. SIGNET: Efficient neural representation for light fields. In: 2021 IEEE/CVF international conference on computer vision (ICCV). Montreal, QC, Canada: IEEE; 2021, p. 14204–13. http://dx.doi.org/10.1109/ICCV48922.2021.01396.

[60] Attal B, Huang J-B, Zollhofer M, Kopf J, Kim C. Learning neural light fields with ray-space embedding. In: 2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR). New Orleans, LA, USA: IEEE; 2022, p. 19787–97. http://dx.doi.org/10.1109/CVPR52688.2022.01920.

[61] Renteln P. Manifolds, tensors, and forms: an introduction for mathematicians and physicists. Cambridge University Press; 2013.

[62] Zhou Y, Barnes C, Jingwan L, Jimei Y, Hao L. On the continuity of rotation representations in neural networks. In: The IEEE conference on computer vision and pattern recognition (CVPR). 2019.

[63] Sitzmann V, Rezchikov S, Freeman B, Tenenbaum J, Durand F. Light field networks: Neural scene representations with single-evaluation rendering. Adv Neural Inf Process Syst 2021;34:19313–25.

[64] Tancik M, Srinivasan PP, Mildenhall B, Fridovich-Keil S, Raghavan N, Singhal U, et al. Fourier features let networks learn high frequency functions in low dimensional domains. 2020.

[65] Mukund VT, Wang P, Chen X, Chen T, Venugopalan S, Wang Z. Is attention all that NeRF needs? 2023, http://dx.doi.org/10.48550/arXiv.2207.13298.

[66] Neff T, Stadlbauer P, Parger M, Kurz A, Mueller JH, Chaitanya CRA, et al. DONeRF: Towards real-time rendering of compact neural radiance fields using depth oracle networks. Comput Graph Forum 2021;40:45–59. http://dx.doi.org/10.1111/cgf.14340.

[67] Yenamandra T, Tewari A, Yang N, Bernard F, Theobalt C, Cremers D. FIRe: Fast inverse rendering using directional and signed distance functions. 2022, http://dx.doi.org/10.48550/arXiv.2203.16284.

[68] Jia Y-B. Plücker coordinates for lines in the space. 2020, https://faculty.sites.iastate.edu/jia/files/inline-files/Plücker-coordinates.pdf.

[69] Feng BY, Zhang Y, Tang D, Du R, Varshney A. PRIF: Primary ray-based implicit function. In: Avidan S, Brostow G, Cissé M, Farinella GM, Hassner T, editors. Computer vision – ECCV 2022, Vol. 13663. Cham: Springer Nature Switzerland; 2022, p. 138–55. http://dx.doi.org/10.1007/978-3-031-20062-5_9.

[70] Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks. In: International conference on machine learning. PMLR; 2017, p. 1126–35.

[71] Rusu AA, Rao D, Sygnowski J, Vinyals O, Pascanu R, Osindero S, et al. Meta-learning with latent embedding optimization. In: International conference on learning representations. 2018, p. 11.

[72] Sitzmann V, Chan ER, Tucker R, Snavely N, Wetzstein G. MetaSDF: Meta-learning signed distance functions. Adv Neural Inf Process Syst 2020;33:10136–47.

[73] Kim DH, Yun ID, Lee SU. Graph representation by medial axis transform for 3D image retrieval. In: Three-dimensional image capture and applications IV, Vol. 4298. SPIE; 2001, p. 223–30. http://dx.doi.org/10.1117/12.424910.

[74] He S, Choi Y-K, Guo Y, Guo X, Wang W. A 3D shape descriptor based on spectral analysis of medial axis. Comput Aided Geom Design 2015;39:50–66. http://dx.doi.org/10.1016/j.cagd.2015.08.004.

[75] Lin C, Liu L, Li C, Kobbelt L, Wang B, Xin S, et al. SEG-MAT: 3D shape segmentation using medial axis transform. IEEE Trans Vis Comput Graphics 2022;28:2430–44. http://dx.doi.org/10.1109/TVCG.2020.3032566.

[76] Du H, Qin H. Medial axis extraction and shape manipulation of solid objects using parabolic PDEs. In: Proceedings of the ninth ACM symposium on solid modeling and applications. Goslar, DEU: Eurographics Association; 2004, p. 25–35.

[77] Thiery J-M, Guy É, Boubekeur T. Sphere-meshes:Shape approximation using spherical quadric error metrics. ACM Trans Graph 2013;32:1–2. http://dx.doi.org/10.1145/2508363.2508384.

[78] Thiery J-M, Guy É, Boubekeur T, Eisemann E. Animated mesh approximation with sphere-meshes. ACM Trans Graph 2016;35:1–3. http://dx.doi.org/10.1145/2898350.

[79] Tkach A, Pauly M, Tagliasacchi A. Sphere-meshes for real-time hand modeling and tracking. ACM Trans Graph 2016;35:1. http://dx.doi.org/10.1145/2980179.2980226.

[80] Angles B, Rebain D, Macklin M, Wyvill B, Barthe L, Lewis J, et al. VIPER:Volume invariant position-based elastic rods. Proc ACM Comput Graph Interact Techn 2019;2:1–26. http://dx.doi.org/10.1145/3340260.

[81] Bouix S, Siddiqi K. Divergence-based medial surfaces. In: Computer vision - ECCV 2000. Berlin, Heidelberg: Springer; 2000, p. 603–18. http://dx.doi.org/10.1007/3-540-45054-8_39.

[82] Tam R, Heidrich W. Shape simplification based on the medial axis transform. In: IEEE visualization, 2003. VIS 2003. 2003, p. 481–8. http://dx.doi.org/10.1109/VISUAL.2003.1250410.

[83] Rebain D, Angles B, Valentin J, Vining N, Peethambaran J, Izadi S, et al. LSMAT least squares medial axis transform. Comput Graph Forum 2019;38:5–18. http://dx.doi.org/10.1111/cgf.13599.

[84] Attali D, Boissonnat J-D, Edelsbrunner H. Stability and computation of medial axes - a state-of-the-art report. In: Möller T, Hamann B, Russell RD, editors. Mathematical foundations of scientific visualization, computer graphics, and massive data exploration. Berlin, Heidelberg: Springer; 2009, p. 109–25. http://dx.doi.org/10.1007/b106657_6.

[85] Yang B, Yao J, Wang B, Hu J, Pan Y, Pan T, et al. P2MAT-NET: Learning medial axis transform from sparse point clouds. Comput Aided Geom Design 2020;80:101874. http://dx.doi.org/10.1016/j.cagd.2020.101874.

[86] Tagliasacchi A, Delame T, Spagnuolo M, Amenta N, Telea A. 3D skeletons: A state-of-the-art report. Comput Graph Forum 2016;35:573–97. http://dx.doi.org/10.1111/cgf.12865.

[87] Igehy H. Tracing ray differentials. In: Proceedings of the 26th annual conference on computer graphics and interactive techniques - SIGGRAPH '99. Not Known: ACM Press; 1999, p. 179–86. http://dx.doi.org/10.1145/311535.311555.

[88] Cohen-Steiner D, Morvan J-M. Restricted delaunay triangulations and normal cycle. In: Proceedings of the nineteenth annual symposium on computational geometry. New York, NY, USA: Association for Computing Machinery; 2003, p. 312–21. http://dx.doi.org/10.1145/777792.777839.

[89] Ba JL, Kiros JR, Hinton GE. Layer normalization. 2016.

[90] Ben-Shabat Y, Koneputugodage CH, Gould S. DiGS: Divergence guided shape implicit neural representation for unoriented point clouds. 2021.

[91] He K, Zhang X, Ren S, Sun J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In: 2015 IEEE international conference on computer vision (ICCV). Santiago, Chile: IEEE; 2015, p. 1026–34. http://dx.doi.org/10.1109/ICCV.2015.123.

[92] The stanford 3D scanning repository. Stanford Computer Graphic Laboratory Homepage; 2014, https://graphics.stanford.edu/data/3Dscanrep/ (accessed January 27, 2023).

[93] Wang Y, Asafi S, van Kaick O, Zhang H, Cohen-Or D, Chen B. Active co-analysis of a set of shapes. ACM Trans Graph 2012;31:165. http://dx.doi.org/10.1145/2366145.2366184, 1–0.

[94] Kleineberg M, Fey M, Weichert F. Adversarial generation of continuous implicit shape representations. 2020.

[95] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in python. J Mach Learn Res 2011;12:2825–30.

[96] Kingma DP, Ba J. Adam: A method for stochastic optimization. 2017.

[97] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. Pytorch: An imperative style, high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A, Alché-Buc Fd', Fox E, Garnett R, editors. Advances in neural information processing systems, Vol. 32. Curran Associates, Inc.; 2019, p. 8024–35.

[98] William F. The PyTorch lightning team. In: PyTorch lightning. 2019, http://dx.doi.org/10.5281/zenodo.3828935.

[99] Själander Magnus, Jahre Magnus, Tufte Gunnar, Reissmann Nico. EPIC: An Energy-Efficient, High-Performance GPGPU Computing Research Infrastructure. 2022, http://dx.doi.org/10.48550/arXiv.1912.05848.

[100] Scopatz A. Pyembree: Python wrapper for intel embree 2.17.7. 2022.

[101] Ravi N, Reizenstein J, Novotny D, Gordon T, Lo W-Y, Johnson J, et al. Accelerating 3D deep learning with PyTorch3D. 2020.

[102] Barré-Brisebois C, Bouchard M. Approximating translucency for a fast cheap and convincing subsurface scattering look. In: Game developers conference, Vol. 6. 2011.

[103] Ward GJ. Measuring and modeling anisotropic reflection. In: Proceedings of the 19th annual conference on computer graphics and interactive techniques. 1992, p. 265–72.